МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
„ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

WEB-ПРОЕКТУВАННЯ

Практикум

для студентів комп’ютерних спеціальностей

Затверджено
редакційно-видавничою
радою університету,
протокол № 13 від 25. 06. 2015 року

Харків 2016

УДК 004.738.1 www
ББК 32.973-018
З-96

Рецензенти:
В. М. Кошельник, д-р. техн. наук, проф. НТУ “ХПІ” (м. Харків),
Г.Ф. Кривуля, д.-р. техн. наук, проф. ХНУРЕ (м. Харків)

Публікується за рішенням вченої ради університету,
Протокол №13 від 25. 06. 2015 р.

З-96 **Web-проектування**: Практикум / І.С. Зиков, О.А., О.П. Черних. – Харків: НТУ “ХПІ”, 2016. – 184 с.

Даний практикум містить 14 лабораторних робіт. У кожній роботі є стислі методичні вказівки щодо підготовки до роботи, приклади виконання HTML-сторінок з використанням стилів CSS та мови JavaScript для виконання індивідуального завдання. Наприкінці методичних вказівок наведено список літератури, рекомендованої до самостійної роботи.

Призначений студентам спеціальностей 7.091501 «Комп’ютерні системи та мережі», 7.091502 «Системне програмування», 7.091503 «Спеціалізовані комп’ютерні системи» бакалаврата 6.0915 «Комп’ютерна інженерія».

Іл. 22. Табл. 19. Бібліогр. 10 назв.

УДК 004.738.1 www
ББК 32.973-018

ВСТУП

Практикум націлено на закріплення теоретичних знань та набуття практичних навичок зі створення HTML-сторінок з використанням стилів CSS та мови JavaScript.

Даний практикум містить 14 лабораторних робіт. У кожній роботі є стислі методичні вказівки щодо підготовки до роботи, приклади виконання HTML-сторінок для виконання індивідуального задання. Наприкінці лабораторного практикуму наведено список літератури, рекомендованої до самостійної роботи.

У результаті виконання лабораторних робіт студенти повинні навчитися: оперувати основними поняттями та загальними принципами мов HTML і JavaScript; форматувати текст Web-сторінок засобами HTML і CSS; розміщувати таблиці, зображення, списки; використовувати фрейми, обробляти Web-форми; обробляти події і змінювати властивості об'єктів на Web-сторінках; керувати елементами на Web-сторінках.

ЛАБОРАТОРНА РОБОТА 1

ФОРМАТУВАННЯ ТЕКСТУ НА WEB–СТОРІНЦІ ЗАСОБАМИ HTML

Мета: вивчення засобів і можливостей HTML для задання обсягу, кольору, стилю та інших параметрів тексту на Web-сторінці.

Загальні положення

У 1960 році американський вчений Тед Нельсон (Ted Nelson) поставив перед собою амбітну мету: об'єднати в єдиній комп'ютерній мережі все більш-менш значущі текстові документи, створені до того часу людством, і логічно зв'язати їх між собою. При цьому читач міг з будь-якого місця одного документа переходу до інших, що пояснює або містить додаткову інформацію.

У 1965 році Нельсон назвав такий метод організації текстової інформації гіпертекстом. Цей проект, який був названий Xanudu, так і не був реалізований в силу отсутсвия та той момент як апаратних і мережевих засобів, так і необхідних інформаційних технологій. З появою і широким використанням глобальної комп'ютерної мережі Internet становище змінилося. І коли в Європейській раді з ядерних досліджень CERN (фр. Consel Europen pour la Rechercher Nucleaire) в Женеві (Швейцарія) була поставлена задача об'єднати електронні документи організації в одне ціле, то для доступу до них було вирішено використовувати Internet. У 1989 році, вирішуючи це задання, співробітник CERN англійський вчений Тім Бернерс-Лі (Tim Berners-Lee) розробив мову створення платформонезавісімих, пов'язаних один з одним текстових документів, котрий був названий HTML (HyperText Markup Language - мова розмітки гіпертекста).

Для перегляду гіпертекстових документів Бернерс-Лі в 1990 році розробив програму − Web-браузер або просто браузер (від англ. to browse − вистежувати). У літературі цю програму називають також Internet-браузером. Цій програмі він дав ім'я WordWideWeb, звідки пішли такі назви, як "WWW" ("Word Wide Web"), "Web" і "Всесвітня павутина". WWW зараз визначається як глобальна інформація, яку иожно читати і змінювати через комп'ютер, підключений до мережі Internet.

У тому ж році Бернерс-Лі запропонував URL (Uniform Resource Locator) − уніфікований локатор (визначник місцезнаходження) ресурсу (раніше називався Universal Resource Locator − універсальний локатор ресурсу). Зараз URL застосовується для позначення адрес майже всіх ресурсів Інтернету.

У 1991 році Тім Бернерс-Лі розробив протокол HTTP (англ. HyperText Transfer Protocol − протокол передачі гіпертексту), як механізм для доступу до документів в Інтернеті і полегшення навігації за допомогою використання гіпертекста.

У 1993 році американський студент Марк Андерссен створив браузер Mosaic, який мав графічний інтерфейс, працював з мишею і фактично став індустріальним стандартом у цій галузі. Пізніше М. Андерссен заснував фірму Nescape.

1.1. Основні поняття HTML

1.1.1. Мова розмітки гіпертекста HTML

Як це випливає з назви, HTML не є алгоритмічною мовою програмування, а, як мова розмітки, володіє засобами подання тексту і зображень на екрані у вигляді Web-сторінки. HTML також дозволяє працювати з гіпертекстовими посиланнями, за допомогою яких здійснюється доступ до інших Web-сторінок.

Стандарт HTML розроблен і підтримується некомерційною міжнародною організацією W3C (World Wide Web Consortium), яка заснована в 1994 році, директором якої нині є Т. Бернерс-Лі. Остання версія HTML має номер 4.01 (прийнята 24.12.1999 року).

Оскільки HTML-документ записується в ASCII-форматі, для створення Web-сторінки може бути використаний будь-який текстовий редактор, що працює з простим текстом, тобто таким текстом, який не містить службової інформації. Такими редакторами є, наприклад, Блокнот (NotePad) або FAR.

Після того як за допомогою текстового редактора HTML-документ створений, він повинен бути збережен на диску у вигляді файла з розширенням .htm або .html. Це дозволить після клацання миші по цьому файла запустити браузер, який здійснить формування Web-сторінки відповідно до цього HTML-документа. Можна вчинити інакше. Спочатку

запустити браузер, а потім у його полі адреси (Address) вказати шлях до файла з даними HTML-документа.

Основними функціями браузера є:

- встановлення за допомогою протоколу HTTP по введеною користувачем адресою Web-сторінки з'єднання з Web-сервером, передача йому запиту та отримання від нього необхідної Web-сторінки у вигляді HTML-документа;
- перетворення отриманого HTML-документа в зображення на екрані користувача, яке є власне Web-сторінкою (якщо HTML-документ знаходиться не на Web-сервері, а на комп'ютері користувача, то в якості адреси Web-сторінки необхідно вказати шлях до файла з HTML- документом і в цьому випадку з'єднання з Web-сервером не виконується).

Прикладами браузерів можуть бути такі програми, як Microsoft Internet Explorer (MIE), Mozilla Firebox, Apple Safari, Netscape Navigator, Opera, Google Chrome та ін.

За допомогою HTML можна створювати тільки статичні Web-сторінки. Для створення динамічних сторінок необхідно спільне використання HTML з такими мовами, як JavaScript, PHP та ін.

1.1.2. Протокол HTTP

Основою HTTP є технологія "клієнт-сервер", тобто передбачається існування споживачів (клієнтів), які ініціюють з'єднання і посилають запит, і постачальників (серверів), які очікують з'єднання для отримання запиту, роблять необхідні дії і повертають назад повідомлення з результатом. HTTP в даний час повсюдно використовується в мережі Internet для отримання інформації з Web-сайтів. HTTP використовується також в якості "транспорту" для інших протоколів прикладного рівня, таких як SOAP, XML-RPC, WebDAV.

Основним об'єктом маніпуляції в HTTP є ресурс, на який вказує URI (англ. Uniform Resource Identifier) в запиті клієнта. Зазвичай такими ресурсами файли, які зберігаються на сервері, але ними можуть бути логічні об'єкти або щось абстрактне.

Особливістю протоколу HTTP є можливість вказати в запиті і відповіді спосіб подання одного і того ж ресурсу з різними параметрами: формата, кодування, мови і т. д. Саме завдяки можливості вказівки способу кодування

повідомлення клієнт і сервер можуть обмінюватися двійковими даними, хоча даний протокол є текстовим.

HTTP − протокол прикладного рівня, аналогічними йому є FTP і SMTP. Обмін повідомленнями йде по звичайній схемі "запит-відповідь". Для ідентифікації ресурсів HTTP використовує глобальні URI. На відміну від багатьох інших протоколів, HTTP не зберігає свого стану. Це означає відсутність збереження проміжного стану між парами "запит-відповідь". Компоненти, що використовують HTTP, можуть самостійно здійснювати збереження інформації про стан, пов'язаної з останніми запитами і відповідями. Браузер, що посилає запити, може відстежувати затримки відповідей. Сервер може зберігати IP-адреси і заголовки запитів останніх клієнтів. Протоколу, який неінформований про попередні запити і відповіді, не передбачена внутрішня підтримка стану, до нього не висуваються такі вимоги.

Найбільш рання версія протоколу HTTP/0.9 була вперше опублікована в січні 1992 р. Специфікація протоколу привела до впорядкування правил взаємодії між клієнтами і серверами HTTP, а також чіткому розподілу функцій між цими двома компонентами. Були задокументовані основні синтаксичні та семантичні положення.

Поточна версія HTTP/1.1 прийнята в червні 1999 року. Новим у цій версії був режим "постійного з'єднання": TCP-з'єднання може залишатися відкритим після відправлення відповіді на запит, що дозволяє посилати кілька запитів за одне з'єднання. Клієнт тепер зобов'язаний надсилати інформацію про ім'я хоста, до якого він звертається, що зробило можливою більш просту організацію віртуального хостингу.

1.1.3. Визначення ресурсів URL

Стандарт URL закріплений у документі RFC 1738 і регулюється організацією IETF (Internet Engineering Task Force), заданням якої яляется просування стандартів Internet. Спочатку локатор URL був розроблений як система для максимально природного зазначення на місцезнаходження ресурсів в мережі. Локатор повинен був бути легко розширюваним і використовувати лише обмежений набір ASCII символів. У зв'язку з цим виникла наступна традиційна форма запису URL:

<Схема>://<логін>:<пароль>@<хост>:
<порт>/<шлях>?<Параметри> #<якір>, де:

- схема − схема звернення до ресурсу, в більшості випадків мається на увазі мережевий протокол;
- логін − ім'я користувача, що використовується для доступу до ресурса;
- пароль − пароль зазначеного користувача;
- хост − повністю прописане доменне ім'я хоста в системі DNS або IP-адреса хоста у формі чотирьох десяткових чисел, розділених крапками; числа − натуральні в інтервалі від 0 до 255;
- порт − порт хоста для підключення;
- шлях – інформація про місце знаходження ресурса; залежить від протокола;
- параметри − рядок запиту з переданими на сервер (методом GET) параметрами;
- якір − ідентифікатор якоря (внутрішнього посилання), що посилається на деяку частину (розділ) відкриваючого документа.
- Загальноприйняті схеми (протоколи) URL включають:
- ftp − протокол передачі файлів FTP;
- http − протокол передачі гіпертекста;
- https − спеціальна реалізація протоколу HTTP, що використовує шифрування (як правило, SSL або TLS);
- gopher − протокол Gopher;
- mailto − адреса електронної пошти;
- news − новини Usenet;
- irc − протокол IRC;
- prospero − служба каталогів Prospero Directory Service;
- telnet − посилання на інтерактивну сесію Telnet;
- wais − база даних системи WAIS;
- xmpp − протокол XMPP (частина Jabber);
- skype − протокол Skype.

В даний час URL позиціонується як частина більш загальної системи ідентифікації ресурсів − URI (Uniform Resource Identifier) − уніфікований індефікатор ресурсу, який являє собою символьний рядок, що дозволяє

ідентифікувати який-небудь ресурс: документ, зображення, файл, службу, адрес електронної пошти та ін.

1.2. Структура HTML-документа

HTML-документ складається з слів двох типів: службових слів, які на екран не виводяться, та власне тексту, який виводиться на екран. Службові слова називаються тегами (або тегами) від англ. tags і беруться в кутові дужки <>. При відображенні HTML-документа на екран тегам відповідають елементи Web-сторінки. Теги можуть мати параметри (атрибути), які задають (змінюють) окремі властивості елементів, наприклад, розмір, колір, вирівнювання та ін. Параметри вказуються всередині тега після його найменування і відділяються друг від друга і від найменування тега пробілами.

Теги бувають подвійними, які називають контейнерами (container), і одиночними. Прикладом контейнера служать теги, які задають початок HTML-документа: <HTML> і його закінчення: </HTML>. Тег закінчення контейнера відрізняється від тега його початку наявністю символу "/". Текст, що знаходиться між початком контейнера та його закінченням, є його вмістом.

Прикладом одиночного тега служить тег перекладу на новий рядок: <BR>. Його застосування обумовлене тим, що браузери перетворять повледовательності пробілів, а також символи переведення каретки (CR) і переходу на новий рядок (LF), які є у HTML-документі, створеному текстовим редактором, в один символ пропуску, виведений на екран.

У HTML-документі в найменуваннях тегів та їх параметрів (атрибутів), а також у словах, які є їх значеннями, великі і малі літери не розрізняються. Надалі будемо дотримуватися наступної угоди: найменування тегів і параметрів будуть вказуватися головними літерами, а їх значення – малими літерами.

Згідно з вимогами організації World Wide Web Consortium, щоб бути правильно оформленим, HTML-документ повинен починатися з тега прологу <!DOCTYPE>, за яким іде основна частина HTML-документа (<HTML>... </HTML>), що складається з заголовка HTML-документа (<HEAD>... </HEAD>) і тіла HTML-документа (<BODY>... </BODY>).

1.2.1. Тег пролога

Тег пролога задає тип поточного документа − DTD (document type definition − опис типу документа). Для HTML дозволені наступні три його варіанти:

- документ, суворо підтримує специфікації W3C –
- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
- перехідний документ –
- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
- використовує фрейми –
- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" http://www.w3.org/TR/html4/frameset.dtd">`

Використання тега пролога наведено в ***прикладі 1.1.***

1.2.2 Заголовок HTML-документа

У заголовку HTML-документа, який знаходиться всередині тега-контейнера <HEAD>. . . </HEAD>, міститься інформація, яка не виводиться в основне вікно Web-сторінки. Вона використовується браузером для інших цілей. Наприклад, за допомогою контейнера <TITLE>. . . </TITLE> може бути заданий титул Web-сторінки, що поміщається наверху екрана. Його зміст указується в переліку відвіданих сторінок і в інших місцях. Якщо розробник Web-сторінки хоче, щоб його сайт був затребуваний, йому необхідно уважно поставитися до змісту титулу Web-сторінки. Дана Web-сторінка при використанні браузера MIE має такий титул:

```
Лабораторна робота 1: форматування тексту на Web-сторінці засобами HTML - Microsoft Internet Explorer
```

Останні три слова додає сам браузер.

Крім того заголовок HTML-документа може містити описи каскадних листків стилів (лабораторна робота 2 "Форматування текста на Web-сторінці засобами CSS") або фрагменти програми на JavaScript (лабораторна робота 9

"Основи мови JavaScript"). Використання тега <TITLE> в заголовку HTML-документа наведено в ***прикладі 1.2***.

1.2.3. Тіло HTML-документа

Зміст тіла HTML-документа, що знаходиться всередині тега-контейнера <BODY>. . . </BODY>, містить дані (текст, зображення, посилання тощо), які будуть безпосередньо виводитися на екран у вікно браузера.

Прикладом найпростішого HTML-документа, який виводить на екран традиційне повідомлення "Привет, мир!", є наступний текст:

Приклад 1.1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<BODY>
Привет, мир!
</BODY>
</HTML>
```

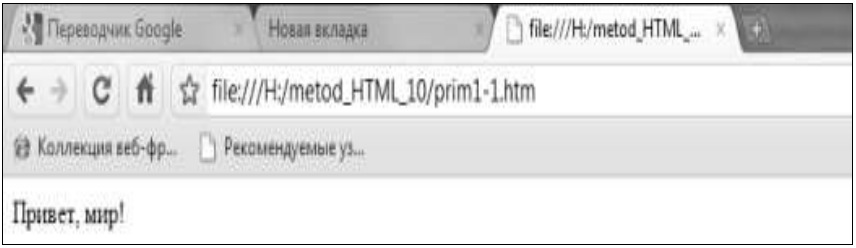


Рисунок 1.1 – Результат прикладу 1.1 на екрані

У цьому прикладі використаний тег прологу, який показує, що цей HTML-документ є перехідним і не відповідає вимогам польностью W3C. Враховуючи, що більшість HTML-документів в WWW не мають цього тега, та з метою зменшення обсягу HTML-документів тег прологу не буде використовуватися в подальшому в прикладах лабораторних робіт даного практикуму.

Однак треба пам'ятати, що при наявності вимог з оформлення HTML-документа в строгій відповідності зі специфікаціями W3C, включення тега прологу в HTML-документ є обов'язковим.

Результат виконання ***прикладу 1.1*** – повідомлення "Привет, мир!", яке знаходиться у верхньому лівому кутку екрана, має чорний колір, невеликий розмір літер (ці параметри задані за замовчуванням) і виглядає непривабливо. Щоб зробити його вигляд більш приємним і легким для читання, необхідно виконати форматування тексту, тобто задати його колір, обсяг, вирівнювання та інші параметри засобами HTML (див. розділ 1.2).

Щоб побачити вихідний текст поточної Web-сторінки, необхідно на панелі керування браузера вибрати "Вид" ("View"), а потім "Перегляд HTML-коду" ("Source").

Зазначимо, що сучасні браузери здатні розуміти і спрощену структуру документа. Наприклад, HTML-документ, що складається з одного рядка:

```
Привет, мир! ,
```

дасть на екрані таке ж повідомлення. Але ми надалі будемо дотримуватися повної форми HTML-документа.

1.2.4. Задання коментарів

Для вказівки одно- і багаторядкових коментарів в HTML-документі використовується така конструкція:

```
<! –
Цей текст браузером не обробляється.
Він закоментований.
->
```

Як коментар також можуть бути використаний тег-контейнер <COMMENT>. . . </COMMENT>.

Зазначимо, що теги початку і закінчення коментаря поміщаються в HTML-документі тільки між тегами HTML-документа, тобто закоментувати окремі параметри тега можна. Зразок використання коментаря наведено в ***прикладі 1.2***.

2.5. Види елементів Web-сторінки

Поточний стандарт HTML передбачає три види елементів Web-сторінки:

- блоковий елемент (або блок), який відображається на екрані окремо від інших елементів додаванням перед змістом і після змісту елементу порожнього рядка. Прикладом блоку може служити параграф (див. підрозділ 1.3.3);
- позиція списку, ведуча аналогічно блочному елементу, але має маркер списку (див. лабораторну роботу 3 "Списки на Web-сторінці");
- вбудований елемент, який відображається всередині блоку в загальному текстовому потоці. Прикладом вбудованого елементу може бути зображення (див. лабораторну роботу 5 "Графіка на Web-сторінці").

Вид елемента може бути змінений (див. підрозділ 14.1.1 лабораторної роботи 14 "Приховування елементів. Робота з клавіатурою").

1.3. Форматування тексту за допомогою HTML

1.3.1. Задання полів на Web-сторінці

Для задання лівого, правого, верхнього та нижнього поля документа на Web-сторінці використовуються відповідно параметри LEFTMARGIN, RIGHTMARGIN, TOPMARGIN і BOTTOMMARGIN тега <BODY>. Значення кожного поля вказується в пікселях. Наприклад, верхнє поле розміром 50 пікселів задається так:

```
<BODY TOPMARGIN=50>
```

У ***прикладі 1.2*** наведено задання розміру лівого поля Web-сторінки.

1.3.2. Задання заголовків Web-сторінки

В HTML-документі може бути використано до шести заголовків різного рівня. Вони задаються за допомогою тегів-контейнерів:

- <H1> Назва заголовка самого високого рівня </ H1>;
- <H2>. . . </ H2>;
- <H3>. . . </ H3>;
- <H4>. . . </ H4>;

- <H5>. . . </ H5>;
- <H6> Назва заголовка найнижчого рівня </ H6>.

1.3.3. Задання параграфів

Тіло HTML-документа може бути поділено на параграфи (окремі текстові блоки). Це робиться з метою наочності та / або для задання усьому текстовому блоку певних параметрів. Для зазначення параграфа використовується контейнер <P>. . . </P>:

```
<P>
Текст
параграфа
</P>
```

Оскільки тег <P> є блоковим елементом, на екрані додається новий рядок на початку і в кінці змісту тега. Тег закінчення параграфа </P> можна не вказувати. В цьому випадку тег початку наступного параграфа <P> означатиме закриття поточного параграфа (а тег </BODY> - останнього). Однак слід пам'ятати, що згідно з рекомендаціями W3C кожен тег-контейнер повинен бути закритий. Застосування параграфів наведено в ***прикладі 1.2***.

1.3.4. Вирівнювання тексту на Web-сторінці

За замовчуванням заголовки Web-сторінки і параграфи вирівняні горизонтально по лівому краю вікна бразуера. Для того, щоб змінити це начальне вирівнювання і зробити Web-сторінку більш зручною для перегляду, необхідно використовувати параметр ALIGN. Він може приймати такі значення:

- left – текст вирівняно по лівому краю (задається за замовчуванням);
- right – текст вирівняно по правому краю;
- center – текст вирівняно по центру;
- justify – текст вирівняно по ширині екрана.

Наприклад:

```
<P ALIGN=right> Текст вирівняно по правому краю
```

Текст також може бути також вирівняно по центру за допомогою тега-контейнера <CENTER>. . . </CENTER>:

```
<CENTER> Текст вирівняно по центру </CENTER>
```

У цьому випадку текст між тегами <CENTER> і </ CENTER> виводиться по центру нового рядка екрана. У ***прикладі 1.2*** приведені усі варіанти вирівнювання текста.

1.3.5. Задання кольору на Web-сторінці

Коли браузер виводить на екран текст, то як колір тексту та колір фону він використовує значення, задані за замовчуванням (у ***прикладі 1.1*** колір тексту – чорний, а колір фону – світло-сірий). Проте в HTML є засоби для задання і зміни як кольору тексту, так і кольору фону. Для цього використовуються відповідно параметри TEXT і BGCOLOR тега <BODY>.

Якщо необхідно вказати деякий колір тільки для частини тексту, то використовується параметр COLOR тега <FONT>.

В HTML є два способи задання значення кольору.

У першому способі значення кольору задаються за допомогою трьох складових кольорів у вигляді чисельного значення # RrGgBb, де:

- Rr – дві 16-кові цифри, які визначають компоненту червоного кольору;
- Gg – дві 16-кові цифри, які визначають компоненту зеленого кольору;
- Bb – дві 16-кові цифри, які визначають компоненту синього кольору.

Значення компоненти кожного кольору може мінятися від 0 до FFh, тобто від 0 до 255. Наприклад, значення #FF0000 задає яскраво-червоний колір, #00AA00 – темно-зелений, #C0C0C0 – сріблястий. У такий спосіб можна задати більше 16 млн (2^{24}) кольорів.

Щоб визначити чисельне значення кольору можна скористатися палітрою, яка виводиться на екран після натискання кнопки "Палітра". Після клацання по будь-якому кольору палітри на екран виводиться чисельне значення вибраного кольору.

Другим способом можна задати тільки невелике число основних кольорів у вигляді символічних (англійських) назв. У таблиці 1.1 наведені ці кольори, їх символічні назви та чисельні значення у вигляді #RrGgBb.

Таблиця 1.1 – Символічні назви і чисельні значення кольорів

Колір	Символічна назва	Чисельне значення
Морської хвилі	Aqua	#00FFFF
Чорний	Black	#000000
Блакитний	Blue	#0000FF
Темно-червоний	Futsia	#FF00FF
Сірий	Gray	#808080
Зелений	Green	#008000
Яскраво-зелений	Lime	#00FF00
Темно-червоний	Maroon	#800000
Темно-синій	Navi	#000080
Оливковий	Olive	#808000
Пурпурний	Purple	#800080
Червоний	Red	#FF0000
Срібний	Silver	#C0C0C0
Темної морської хвилі	Teal	#008080
Білий	White	#FFFFFF
Жовтий	Yellow	#FFFF00

Зразки задання кольору на Web-сторінці засобами HTML наведені в ***прикладі 1.2***.

1.3.6. Задання розміру шрифту

Для задання розміру шрифту на Web-сторінці використовується параметр SIZE тега-контейнера <FONT>. . . </FONT>. Текст, зазначений між тегами <FONT> і </FONT>, матиме потрібний розмір. Обсяг попереднього тексту, а також тексту, що йде далі, буде братися за замовчуванням.

Значення параметра SIZE задаються не в пунктах (1 пункт дорівнює 1/72 дюйма), а у відносних одиницях від 1 до 7. Співвідношення між цими

одиницями і пунктами наведено в таблиці 1.2. Розмір за замовчуванням дорівнює 3.

Таблиця 1.2 – Співвідношення між одиницями і пунктами

Відносні одиниці	Пункти (pt)
1	9
2	10
3	12
4	14
5	18
6	24
7	36

Наприклад,

```
<FONT SIZE=5> Розмір шрифту 18 pt </FONT>
```

Можна зробити й так: задати за допомогою тега <BASEFONT> з параметром SIZE базовий розмір шрифту. А потім за допомогою тега <FONT> з параметром SIZE, який набуває значення від -4 до +4, змінити базовий розмір на необхідну величину.

Наприклад,

```
<BASEFONT SIZE=4>
<FONT SIZE=+2> Розмір шрифту 24 pt </FONT>
```

За допомогою тегів-контейнерів <BIG>. . . </BIG> і <SMALL>. . . </SMALL> можна задати текст відповідно великого і малого обсягу, не вказуючи конкретно величину шрифту. Цю величину задає сам браузер.

1.3.7. Задання шрифту на ім'я

Оскільки невідомо, який тип шрифтів використовує браузер на боці користувача, можна за допомогою параметра FACE тега <FONT> явно указати тип імені шрифту.

В одному параметрі FACE можна вказати кілька типів шрифтів, які перераховуються через кому:

```
<FONT FACE="Arial Cyr, Arial, Helv DL"> Шрифт гельветіка
```

Якщо на боці користувача встановлено перший із зазначених шрифтів – Arial Cyr, то рядок "Шрифт Гельветіка" буде відображатися на екрані з використанням саме цього шрифту. Інакше буде зроблена спроба відобразити рядок шрифтом Arial, а потім – Helv DL. Якщо виявиться, що в системі користувача немає жодного з перерахованих шрифтів, рядок буде відображений тим шрифтом, який встановлений в браузері для використання за замовчуванням.

Задзначимо, що значення параметра FACE взяті в подвійні лапки, оскільки імена шрифтів містять пробіли. Але в HTML не буде помилкою вказувати значення будь-якого параметра в лапках. Тому, якщо є сумніви в заданні значень параметрів, краще вказати їх в лапках.

У ***прикладі 1.2*** задані ліве поле, колір і вирівнювання текста, типи і розміри шрифту на Web-сторінці засобами HTML.

Приклад 1.2

```
<HTML>
<HEAD>
<TITLE> HTML: выравнивание текста, задание цвета,
размера и типа шрифта </TITLE>
</HEAD>
<!-- Цвет текста: синий, цвет фона - светло-зеленный: --
>
<BODY LEFTMARGIN="25" TEXT=blue BGCOLOR= #E0FFE0>
<H1 ALIGN=center>ЗАГОЛОВОК 1 УРОВНЯ</H1>
<P ALIGN=right>
<FONT COLOR=red>
<SMALL> Текст малого размера расного цвета выравнен по
правому краю экрана </SMALL>
</FONT>
<P> Выравнивание текста, размер (12 pt) и тип шрифта,
заданы по умолчанию
<P ALIGN=justify> <FONT SIZE=6 COLOR=#FF00FF
FACE="Courier, Courier New Cyr, Courier New"> Текст
```

```
выравнен по ширине экрана с использованием шрифта
Courier лилового цвета размером 24 pt
</FONT>
</BODY>
</HTML
```

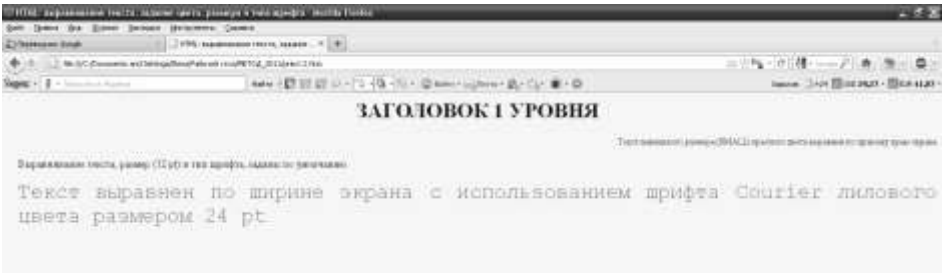


Рисунок 1.2 – Результат прикладу 1.2 на екрані

1.3.8. Фізичне форматування символів

У HTML передбачено кілька тегів-контейнерів для фізичного форматування символів. Ці теги визначають зовнішній вигляд символів явно на противагу тегам логічного форматування (див. підрозділ 1.3.9):

- <B>. . . </B> – виділяє текст жирним шрифтом;
- <I>. . . </I> – виділяє текст похилим шрифтом (курсивом);
- <U>. . . </U> – виділяє текст підкресленням;
- <STRIKE>. . . </ STRIKE> і <S>. . . </ S> – обидва еквівалентних тега виділяють текст перекресленням;
- <TT>. . . </TT> – задає текст з фіксованою шириною символів (наприклад, листкинги програм);
- _{. . .} – вказує підрядковий (нижній) індекс;
- ^{. . .} – вказує надрядковий (верхній) індекс;
- <BLOCKQUOTE>. . . </BLOCKQUOTE> – задає ліве поле;
- <PRE>. . . </PRE> (preformatted – попередньо відформатований) – виводить текст на екран моноширінним шрифтом з відпрацюванням усіх знаків пробілів табуляції, перекладу рядка і нового рядка.

1.3.9. Логічне форматування символів

Теги логічного форматування символів, на відміну від тегів фізичного форматування, вказують ділянки тексту, різняться за смисловим змістом, спосіб виділення яких на екрані визначає сам браузер. У HTML використовуються наступні теги-контейнери логічного форматування:

- <CITE>. . . </CITE> – цитата;
- <EM>. . . </EM> (emphasis – наголос) – текст, що має велике значення;
- <STRONG>. . . </STRONG> (сильний) – сильне виділення тексту;
- <SAMP>. . . </SAMP> (sample – зразок) – зразок повідомлень, що виводяться на екран;
- <KBD>. . . </KBD> (keyboard – клавіатура) – текст, що вводиться з клавіатури;
- <CODE>. . . </CODE> (code – код) – фрагмент вихідного тексту;
- <VAR>. . . </VAR> (variable – змінна) – ім'я змінної.

На Web-сторінках іноді вказують абревіатуру деякого терміна або поняття без його розкриття, наприклад, "ОЗУ" замість "Оперативний запам'ятовувальний пристрій". Щоб пошукові системи в мережі Internet могли здійснювати пошук терміна не тільки за його аббреавітурою, але і за повному позначенням, а також в якості підказки для користувача Web-сторінки, використовується тег-контейнер <ACRONYM> абревіатура </ACRONYM> с параметром TITLE, який розкриває значення використовуваної абревіатури. Замість тега <ACRONYM> з тією ж метою можна також використовувати тег <ABBR>.

Зазначемо, що параметр TITLE може бути використаний для будь-якого елемента Web-сторінки. У цьому випадку при наведенні на цей елемент курсору мищкі на екрані з'являється підказка – спливаюче повідомлення, що є значенням параметра TITLE.

У ***прикладі 1.3*** показано застосування тегів <ACRONYM> і <ABBR> для абревіатур МП і НТ.

1.3.10. Вставка спеціальних символів

У HTML передбачено чотири спеціальних символи, які призначені для службових цілей (для виділення тегів і значень їх параметрів). Це символи <, >, & та ". Вони інтерпретуються браузером особливо і вимагають їх заміни

для включення до тексту як звичайних символів. Заміна потрібна і для подання символів права © та реєстрації ®. У таблиці 1.3 наведені ці заміни.

Таблиця 1.3 – Заміни символів

Символ	Заміна
<	&lt
>	&gt
&	&amp
"	&quot

У загальному випадку в HTML-документ можна вставити будь-який символ, знаючи його ASCII-код. Для цього замість символу підставляється така послідовність символів: &#XX, де XX – десяткове значення ASCII-коду символу. Наприклад, для виведення символу ± необхідно вказати ±177.

До спеціальних символів можна віднести і символ "незворотного пробілу" – (no breaking space). Цей символ використовується тоді, коли необхідно помістити на Web-сторінці кілька пробілів поспіль, оскільки браузери замінюють послідовності поспіль пробілів в HTML-документі одним пропуском при виведенні документа на екран.

1.3.11. Виділення тексту горизонтальною лінією

При оформленні Web-сторінок часто використовується такий прийом, як поділ параграфів тексту або інших його частин горизонтальною лінією.

Для того щоб вкласти в HTML-документ горизонтальну роздільну лінію, необхідно використовувати одиночний тег <HR>. Він має кілька параметрів, які задають зовнішній вигляд лінії:

- ALIGN – визначає вирівнювання лінії і має такі значення:
  - left – по лівому краю;
  - right – по правому краю;
  - center – по центру (задані за умовчанням).
- COLOR – задає колір лінії;
- NOSHADE – якщо зазначений цей параметр, лінія зображується без тіні.
- SIZE – висота лінії в пікселях;
- WIDTH – ширина лінії може виражатися або в пікселях, або у відсотках (%) від величини вікна браузера (за замовчуванням дорівнює 2).

У ***прикладі 1.3*** наведено HTML-документ, який здійснює фізичне і логічне форматування символів на Web-сторінці, розділених горизонтальною лінією:

1.4. Використання тега <META>

Тег <META> призначений для розміщення додаткової інформації про Web-сторінку: описи списка ключових слів для пошукових машин, відомостей про програму, в якій була створена Web-сторінка та ін.

Може знаходиться тільки в секції заголовка HTML-документа. Є блоковим одиночним тегом. Обов'язковими параметрами є PTTP-EQUIV або NAME, що задають тип метаданих, і CONTENT, де вказуються самі метадані зазначеного типу.

Параметр HTTP-EQUIV може приймати такі значення:

- description – задає опис Web-сторінки;
- refresh – задає в секундах інтервал оновлення Web-сторінки, яка відображається браузером;
- url – задає адресу, звідки буде завантажуватися Web-сторінка при оновленні (вимагає наявності тега <META> з параметром refresh);
- mimetype – задає тип даних MIME Web-сторінки (завжди text/html);
- charset – задає кодову сторінку, яка була використана при написанні тексту.

Наприклад, такий тег зіставляє браузер оновлювати (перезавантажувати з Web-сервера) поточну Web-сторінку кожні 30 секунд:

```
<META HTTP-EQUIV="refresh" CONTENT="30">
```

Параметр NAME так само як і параметр HTTP-EQUIV задає тип метаданих і може мати такі значення:

- description – позначає опис Web-сторінки;
- generator – задає ім'я програми, яка створила Web-сторінку;
- keywords – задає список ключових слів, розділених комами (ці слова будуть використані пошуковими машинами);
- progID – ідентифікує програму для редагування Web-сторінок;
- robots – вирішує або забороняє індексацію Web-сторінки пошуковими машинами (доступні два значення параметра CONTENT): all і noindex, відповідно дозволяє і забороняє індексацію;
- template – задає ім'я файла шаблону, застосованого при створенні Web-сторінки (використовується разом з progID).

Приклад 1.3

```
<HTML>
<HEAD>
<TITLE> Физическое и логическое форматирование символов
</TITLE>
</HEAD>
<BODY TEXT=000070>
<CENTER>
<FONT SIZE=6>
<P><B> Полужирный шрифт </B>
<P><STRIKE> Перечеркнутый шрифт </STRIKE>
<P><TT> Шрифт с фиксированной шириной символов </TT>
<P> Формула воды: H <SUB> 2 </SUB> O
<HR COLOR=EEEE00 SIZE=5 WIDTH=80%>
<P><CITE> Цитата: "Среди вещей величайшим есть ничто"
</CITE>
<P><FONT COLOR=FF00FF>
<ACRONYM TITLE="Микропроцессоры "> МП </ACRONYM>
фирмы Intel подтвердили эффективность технологии
<ABBR TITLE="HyperThreading"> HT </ABBR>
</CENTER>
</BODY>
</HTML>
```

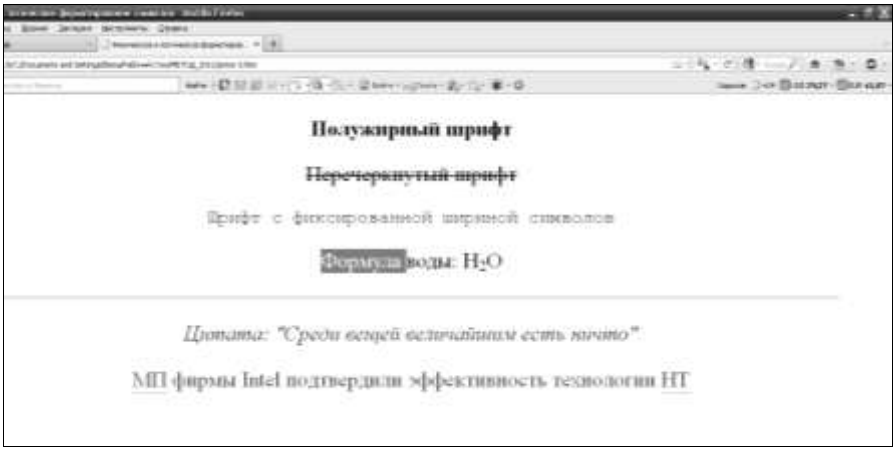


Рисунок 1.3 – Результат прикладу 3 на екрані

Індивідуальні завдання

Оформити реферат на довільну тему у вигляді HTML-документа (Web-сторінки), реалізація якого повинна містити:

- заголовок;
- текст, що складається з параграфів, окремі слова або пропозиції якого мають бути:
  - різних кольорів (не менше трьох);
  - різного обсягу (не менше трьох);
  - подані різними типами шрифтів (не менше двох);
  - мати фізичне і логічне форматування;
  - вирівняні;
  - задані у вигляді абревіатури;
  - прокоментовані;
- мати горизонтальну лінію.

Колір тексту і фону, вигляд фізичного і логічного форматування, а також вид вирівнювання беруться з таблиці 4 відповідно за номером варіанта (номер у журналі). Інші параметри вибираються довільно.

Таблиця 1.4 – Варіанти індивідуальних завдань

№	Колір		Форматування		Вирівнювання
	Тексту	фону	Фізичне	логічне	
1	2	3	4	5	6
1	Синій	світло-зелений	SUP	EM	вправо
2	чорний	світло-жовтий	B	CITE	по центру
3	червоний	світло-синій	I	STRONG	ліве поле, 75 пікселей
4	зелений	світло-червоний	U	VAR	верхне поле, 100 пікселей
5	червоний	світло-зелений	STRIKE	CODE	вправо
6	Синій	світло-жовтий	SUB	SAMP	по центру
7	Білий	Синій	BLOCKQUOTE	KBD	праве поле, 50 пікселей
8	жовтий	чорний	PRE	EM	вправо
9	Синій	світло-сірий	TT	CODE	по центру

Продовження табл. 1.4

1	2	3	4	5	6
10	жовтий	Silver	U	EM	ліве поле, 150 пікселей
11	жовтий	світло-синій	STRIKE	CITE	Вправо
12	зелений	білий	SUB	VAR	по центру
13	Aqua	світло-жовтий	PRE	STRONG	верхнє поле, 40 пікселей
14	Lime	світло-сірий	I	EM	вліво
15	Maroon	Silver	U	CITE	вправо
16	Navi	світло-червоний	SUP	VAR	праве поле, 120 пікселей
17	Purple	світло-зелений	TT	CODE	по центру
18	Silver	Aqua	PRE	CODE	вліво
19	Aqua	світло-червоний	SUB	STRONG	вправо
20	Purple	світло-жовтий	BLOCKQUOTE	CITE	по центру

ЛАБОРАТОРНА РОБОТА 2
ФОРМАТУВАННЯ ТЕКСТУ НА WEB-СТОРІНЦІ ЗАСОБАМИ CSS

Мета: вивчення засобів і можливостей CSS для задання обсягу, кольору, стилю та інших параметрів тексту на Web-сторінці.

Загальні положення

Каскадні листки (таблиці) стилів CSS (Cascading Style Sheets) також, як і мова HTML розроблені консорціумом W3C і теж служать цілям подання документів на Web-сторінках. CSS не є частиною HTML і не використовуються самостійно, але є можливість їх підключення у HTML-документ для спільного застосування.

Каскадні листки стилів, не виконуючи всіх функцій мови HTML, мають розширені можливості при форматуванні тексту. Тому однією із сучасних вимог на розробку Web-сайтів є рекомендація використання CSS замість тегів HTML скрізь, де це можливо.

2.1. Формат CSS

Каскадні листки стилів мають такий формат:

CSS-ідентифікатор (властивість: значення; властивість: значення;. . .).

Як CSS-ідентифікатор можуть бути використані:

- HTML-теги (наприклад, BODY, H1, P, ...);
- HTML-теги з вказівкою класу (наприклад, P.small - для параграфа з маленькими літерами);
- класи без вказівки HTML-тега (наприклад, .green_it - для будь-якого тега, із зеленими похилими символами);
- ідентифікатори (наприклад, #zz).

Нижче наведені властивості каскадних листків стилів і їх можливі значення.

2.1.1. Властивості задання відступів елементів

Властивість margin встановлює величину відступу від кожного краю елемента Web-сторінки. Відступом (полем) є простір від межі поточного

елемента до внутрішньої межі його батьківського елемента. Якщо у елемента немає батька, відступом буде відстань від краю елемента до краю вікна браузера з урахуванням того, що у самого вікна за замовчуванням теж встановлені відступи. Щоб їх позбутися, слід встановлювати значення margin для тега <BODY>, рівним нулю.

Властивість margin дозволяє задати величину відступу відразу для усіх сторін елемента або визначити її тільки для зазначених сторін. Тому може приймати від одного до чотирьох значень:

- якщо задано одне значення – воно застосовується до усіх чотирьох полів елемента;
- якщо задано два значення – перше відноситься до верхнього й нижнього поля, друге – до лівого і правого поля елемента;
- якщо задано три значення – перше відноситься до верхнього поля, друге – до лівого і правого поля, третє – до нижнього поля елемента.
- якщо задано чотири значення – перше відноситься до верхнього поля, друге - до правого поля, третє – до нижнього поля, четверте – до лівого поля елемента.

Значення властивості margin можуть бути задані:

- в одиницях довжини:
  - mm – в міліметрах;
  - cm – в сантиметрах;
  - in – в дюймах (1 дюйм = 2,54 см);
  - px – в пікселах;
  - pt – в пунктах (1 пункт = 1/64 дюйма);
  - pc – в піках (1 пік = 12 пунктів);
- у відсотках від обсягу батьківського елементу;
- словом auto – відступи елемента визначаються браузером (значення за замовчуванням).

Значення відступу елемента можуть бути як позитивними числами, так і негативними. Між числом і одиницею довжини не повинно бути пробілів.

Крім, властивості margin є ще чотири властивості, які описують відступ для кожного поля елемента окремо:

- margin-left – для лівого поля;
- margin-right – для правого поля;

- margin-top – для верхнього поля;
- margin-bottom – для нижнього поля.

Наприклад, CSS

```
P{margin-left:2cm}
```

задає розмір лівого поля всіх параграфів тексту рівним 2 см.

Зазначимо, що у випадку, коли для однієї і тієї властивості тега за допомогою HTML і CSS задаються різні значення, пріоритет віддається CSS. Наприклад, при такому заданні верхнього поля документа

```
<BODY STYLE="margin-top:50" TOPMARGIN=100>
```

верхнє поле матиме розмір 50 пікселів.

2.1.2. Властивості тексту

Для задання параметрів тексту на Web-сторінках використовуються такі властивості CSS:

- text-align – задання горизонтального вирівнювання шрифта;
- text-indent – задання відступу першого рядка параграфа;
- text-decoration – задання декоративного виділення тексту;
- word-spacing – задання відстані між словами у тексті;
- letter-spacing – задання відстані між символами у слові;
- writing-mode – задання напрямків рядків тексту: горизонтальний або вертикальний.

2.1.2.1. Властивість text-align. Горизонтальне вирівнювання тексту на Web-сторінках здійснюється за допомогою властивості text-align, яка має такі значення:

- center – вирівнювання тексту по центру, текст поміщається по центру горизонталі вікна браузера або контейнера, де розташований текстовий блок;
- left – по лівому краю екрана (значення за умовчанням);
- right – по правому краю екрана;

- justify – текст вирівняний по обох краях (якщо довжина тексту більше довжини рядка екрану);
- auto – не змінює положення елемента;
- inherit – успадковує значення батька.

2.1.2.2. Властивість text-indent встановлює величину відступу першого рядка блоку тексту (наприклад, для параграфа <P>). Впливу на усі інші рядки не надає. Допускається від'ємне значення для створення виступу першого рядка, але слід перевірити, щоб текст не виходив за межі вікна браузера. Значення за замовчуванням - 0.

Як значеня приймаються будь-які одиниці довжини, прийняті в CSS (див. підрозділ 2.1.1). При завданні значення у відсотках, відступ першого рядка обчислюється в залежності від ширини блоку.

2.1.2.3. Властивість text-decoration. Крім зазначення кольору та параметрів шрифту, для виділення фрагментів тексту може бути також використана властивість text-decoration, яка може набувати таких значень:

- none – декоративне виділення тексту не використовується (значення за умовчанням);
- underline – підкреслення (лінія знизу);
- overline – надкреслення (лінія зверху);
- line-through – перекреслювання;
- blink – мерехтіння тексту (не підтримується MIE);
- inherit – значення успадковується у батька.

Одночасно можна застосувати більш одного стилю, перераховуючи значення через пробіл.

2.1.2.4 Властивість text-transform керує перетворенням тексту в заголовні або прописні символи. Може набувати таких значень:

- none – не змінює регістр символів (значення за умовчанням);
- capitalize – перший символ кожного слова в реченні буде головним, інші символи свій вигляд не змінюють;
- lowercase – усі символи текста стають рядковими (нижній регістр);
- uppercase – усі символи текста стають прописними (верхній регістр);
- inherit - значення успадковується у батька.

2.1.2.5 Властивість line-height встановлює міжрядковий інтервал текста, відлік ведеться від базової лінії шрифту. За звичайних обставин відстань між рядками залежить від виду та розміру шрифту і визначається браузером автоматично. Негативне значення міжрядкової відстані не допускається. Значення може задаватися:

- у вигляді множника;
- у вигляді одиниць довжини;
- у відсотках;
- словом normal – відстань між рядків обчислюється автоматично.
- словом inherit – успадковує значення батька.

Будь-яке число більше нуля сприймається як множник від розміру шрифту поточного тексту. Наприклад, значення 1.5 встановлює полуторний міжрядковий інтервал. Як значення приймаються також будь-які одиниці довжини, прийняті в CSS (див. підрозділ 2.1.1).

Дозволяється використовувати процентний запис, у цьому випадку за 100% береться висота шрифту.

2.1.2.6. Властивість word-spacing вказує відстань між словами у тексті і може мати такі значення:

- normal – стандартна відстань (значення за умовчанням);
- величина, яка вказує додаткову відстань (одиниці виміру – див. пункт 2.1.1).

2.1.2.7 Властивість letter-spacing указує відстань між символами у слові і може мати такі значення:

- normal – стандартна відстань (значення за умовчанням);
- величина, яка вказує додаткову відстань (одиниці виміру – див. пункт 2.1.1).

Наприклад, CSS

```
.Big_space{letter-spacing:5mm}
```

задає для всіх тегів класу тексту "big_space" зі збільшеною на 5 мм відстанню між літерами.

2.1.2.8. Властивість writing-mode задає напрямок рядків тексту: горизонтальний або вертикальний і може мати такі значення:

- lr-tb – звичайне горизонтальне розташування рядків тексту; текст пишеться зліва направо і зверху вниз (значення за умовчанням);

- tb-rl – повертає текст на 90° за годинниковою стрілкою; при цьому він буде писатися зверху вниз і справа наліво.

2.1.2.9 Властивість white-space встановлює, як відображати пробіли між словами. У звичайних умовах будь-яка кількість пробілів в коді HTML показується на Web-сторінці як один. Винятком є тег <PRE>, у якому текст, поміщений в цей контейнер, виводиться з усіма пробілами, як він був відформатован користувачем. Таким чином, white-space імітує роботу тега <PRE>, але на відміну від нього не змінює шрифт на моноширинний. Може приймати такі значення:

- normal – текст у вікні браузера виводиться як зазвичай, переноси рядків встановлюються автоматично;
- nowrap – пробіли не враховуються, переноси рядків у коді HTML ігноруються, весь текст відображається одним рядком; разом з тим, додавання тега
 переносить текст на новий рядок;
- pre – текст показується з урахуванням усіх пробілів і переносів, як вони були додані розробником в коді HTML. Якщо рядок виходить занадто довгим і не поміщається у вікні браузера, то буде додана горизонтальна смуга прокручування;
- pre-line – в тексті пропуски і переноси не враховуються, текст автоматично переноситься на наступний рядок, якщо він не поміщається в задану область;
- pre-wrap – в тексті зберігаються усі пробіли і перенесення, однак якщо рядок по ширині не поміщається в задану область, то текст автоматично буде перенесений на наступний рядок;
- inherit - успадковує значення батька.

2.1.3. Властивість задання кольору

Для задання кольору на Web-сторінках використовується властивість color. Є п'ять можливостей задання кольору (перші дві такі ж, як у HTML – див. п. 1.2.1 лабораторної роботи 1):

- за допомогою англійського імені кольору;
- за допомогою значення #RrGgBb;
- за допомогою значення #RGB (компоненти змінюються в межах від 0 до 15);

- за допомогою rgb (R, G, B), де кожна складова кольору задається десятковим числом від 0 до 255;
- за допомогою rgb (R%, G%, B%) – аналогічно попередньому, але кожна складова кольору задається у вигляді частки (відсотків).

2.1.4. Властивості задання параметрів шрифтів

Для задання параметрів шрифтів на Web-сторінках використовуються такі властивості CSS:

- font-style - задання стилю шрифту;
- font-weight - задання жирності шрифту;
- font-variant - задання виду малих літер;
- font-size - задання розміру шрифту;
- font-family - задання сімейства шрифтів.

2.1.4.1. Властивість font-style. Властивість стилю шрифту font-style визначає накреслення шрифту - звичайне, курсивом або похиле. Коли для текста встановлено курсивне або похиле накреслення, браузер звертається до системи для пошуку відповідного шрифту. Якщо заданий шрифт не знайдений, браузер використовує спеціальний алгоритм для імітації потрібного виду тексту. Результат і якість при цьому можуть вийти незадовільними, особливо при друку документа. Властивість font-style може приймати такі значення:

- normal – звичайне написання тексту (значення за умовчанням);
- italic – накреслення курсивом;
- oblique – похиле накреслення;
- inherit – успадковує значення батька.

Курсив і похилий шрифт при всій їх схожості не одне і те ж. Курсив це спеціальний шрифт, який імітує рукописний, похилий ж утворюється шляхом нахилу звичайних знаків вправо.

2.1.4.2. Властивість font-weight. Властивість жирності шрифту font-weight має такі значення:

- normal – звичайний (значення за умовчанням);
- bold – напівжирний;
- bolder – жирне накреслення;
- lighter – світле накреслення.

Властивість font-weight можна також задавати в умовних одиницях: 100 | 200 | 300 | 400 (відповідає значенню normal) | 500 | 600 | 700 (відповідає значенню bold) | 800 | 900.

2.1.4.3. Властивість font-variant. Властивість виду малих літер font-variant визначає, як потрібно представляти малі літери - залишити їх без модифікацій або робити їх усі прописними зменшеного розміру. Такий спосіб зміни символів називається капітеллю. Властивість font-variant може приймати такі значення:

- normal – звичайний (значення за умовчанням);
- small-caps – робить малі літери головними, але меншого розміру.

2.1.4.4. Властивість font-size. Розмір шрифту за допомогою властивості font-size може задаватися:

- у вигляді абсолютного розміру шрифта:
  - xx-small - найменший;
  - x-small - дуже маленький;
  - small - маленький;
  - medium - середній (приймається за умовчанням);
  - large - великий;
  - x-large - дуже великий;
  - xx-large – найбільший;
- у вигляді відносного розміру шрифта:
  - smaller - менше;
  - large - більше;
- в одиницях довжини (див. підрозділ 2.1.1);
- у відсотках від розміру батьківського елемента.

Зазначимо, що одиниці задання розміру шрифту em і ex, які рівні відповідно поточному розміру шрифту та розміру літери "x", не рекомендовані до використання.

2.1.4.5. Властивість font-family. Властивість сімейства шрифтів font-family встановлює сімейство шрифту, яке буде використовуватися для оформлення тексту змісту. Список шрифтів може включати одну або кілька назв, розділених комою. Якщо в імені шрифту містяться пробіли, наприклад, Trebuchet MS, воно повинно полягати в одинарні або подвійні лапки.

Коли браузер зустрічає перший шрифт у списку, він перевіряє його наявність на комп'ютері користувача. Якщо такого шрифту немає, береться

наступне ім'я зі списку і також аналізується на присутність. Тому кілька шрифтів збільшує ймовірність, що хоча б один з них буде виявлений на клієнтському комп'ютері. Закінчують список зазвичай ключовим словом, яке описує тип шрифту – serif, sans-serif, cursive, fantasy або monospace. Таким чином, послідовність шрифтів краще починати з екзотичних типів і закінчувати узагальненим ім'ям, яке задає вид накреслення. наприклад:

```
P{font-family: "Arial Cyr", Arial, Helvetica, sans-serif}
```

Якщо властивість font-family визначена у вбудованому підключенні CSS, назви шрифтів беруться не в подвійні кавичні, а в одиночні:

```
<P STYLE="font-family:'Arial Cyr', Arial, Helvetica,
sans-serif">
```

2.1.5. Властивість вертикального вирівнювання

Для вертикального вирівнювання тексту використовується властивість vertical-align, яка може мати наступні значення:

- baseline – вирівнювання базової лінії елемента за базовою лінією родителя (використовується за умовчанням);
- sub – перетворює текст у нижній індекс;
- super – перетворює текст у верхній індекс;
- top – вирівнювання верху елемента по верху батька;
- text-top – вирівнювання верху тексту елемента по верху тексту батька;
- middle – вирівнювання центра елемента по центру батька;
- bottom – вирівнювання низу елемента по низу батька;
- text-bottom – вирівнювання низу тексту елемента по низу тексту батька;
- inherit – успадковує значення батька.

Як значення можна використовувати одиниці довжини (див. підрозділ 2.1.1) і відсотки. Позитивне число зміщує елемент вгору відносно базової лінії, в той час як негативне число опускає його вниз. При використанні відсотків, відлік ведеться від значення властивості line-height, при цьому 0% аналогічно значенню baseline.

2.1.6 Властивості смуг прогруткі. У деяких випадках для додавання Web-сторінці необхідного виду буває корисним змінити вигляд смуг прокручування (scrollbar). Для цього є всі необхідні наступні властивості CSS (повністю підтримуються починаючи з версії 5.5 MIE):

- scrollbar-base-color – задає основний колір смуги прокручування. Якщо не визначено інші властивості смуги прокручування, то бігунок і кнопки прокручування будуть відображатися певним вам кольором із застосуванням тривимірних ефектів. Фон смуги прокручування буде відображатися теж цим кольором, але тільки освітленим;
- для задання більш тонкої настройки смуг прокручування можна застосувати такі властивості CSS:
  - scrollbar-3dlight-color – задає колір верхньої та лівої меж смуги прокручування, її бігунка і стрілок;
  - scrollbar-arrow-color – задає колір стрілок на кнопках смуги прокручування;
  - scrollbar-darkshadow-color – задає колір "тіні", відкидаємо бігунком і кнопками смуги прокручування (колір правих і нижніх граней);
  - scrollbar-face-color – задає основний колір бігунка і кнопок прокручування смуги прокручування;
  - scrollbar-highlight-color – задає колір "освітленій" частини бігунка і кнопок прокручування смуги прокручування (колір лівих і верхніх їх граней);
  - scrollbar-shadow-color – задає колір "неосвітленій" частини бігунка і кнопок прокручування смуги прокручування (колір правих і нижніх їх граней). Не плутати з кольором "тіні", що задається атрибутом scroll-darkshadow-color;
  - scrollbar-track-color – задає колір фону смуги прокручування, тієї її частини, по якій переміщається повзунок.

Крім застосування каскадних листків стилів як засоби форматування тексту CSS, можуть бути використані також для задання:

- властивостей списків (див. розділ 3.2 лабораторної роботи 3 "Розміщення списків на Web-сторінці");
- властивостей таблиці (див. підрозділ 4.1.4 лабораторної роботи 4 "Розміщення таблиць на Web-сторінці");

- властивостей фону, позиціонування елементів на Web-сторінці і смуг прокручування (див. підрозділи 5.1.2, 5.1.3 і приклад 5.1 лабораторної роботи 5 "Розміщення зображень на Web-сторінці");
- властивостей посилань і зображення курсора (див. підрозділи 6.1.2 та 6.1.3 лабораторної роботи 6 "Робота з посиланнями на Web-сторінці").

2.2. Способи задання CSS у HTML-документі

CSS не тільки розширює можливості HTML, але і робить форматування тексту однаковим як для окремо взятого документу, так і для групи документів, які мають загальне призначення. Залежно від того, наскільки загальним буде використання тих чи інших каскадних листків стилів, застосовується один із трьох можливих способів використання CSS у HTML-документі:

- зовнішнє задання CSS;
- внутрішнє задання CSS;
- вбудоване задання CSS.

2.2.1. Зовнішнє задання CSS

При зовнішньому заданні каскадні листки стилів формуються відповідно до описаних вище правил у вигляді окремого файла, так само, як і HTML-документ, за допомогою простого текстового редактора. За розширення файла для зручності користувачів краще вказати .css, хоча можна задавати і інші розширення.

Наприклад, наступний файл з ім'ям my.css

```
H1{text-align:center; color:#D0D; font: italic small-caps 0.4in tahoma}
P{text-indent:1.5cm; color:#000080;font:7mm}
```

містить два каскадних листка стилів, які задають:

❖ параметри заголовка першого рівня (його вирівнювання, колір і вигляд шрифту);

❖ параметри параграфа, які є загальними для HTML-документів (розмір абзацу, колір і розмір шрифту).

Зазначимо, що при описі параметрів заголовка використана скорочена форма задання властивостей шрифту:

```
font:italic small-caps 0.4in tahoma вместо font-style:italic;
font-variant: small-caps;font-size:0.4in; font-family:tahoma
и font:7mm вместо font-size:7mm.
```

Підключення створеного файла CSS до HTML-документа здійснюється за допомогою одиночного тега <LINK>, який розміщується в області заголовка HTML-документа, тобто між тегами <HEAD> і </HEAD>. Тег <LINK> містить такі параметри:

- REL – визначає відносини між поточним документом і файлом, на який посилаються. Це необхідно, щоб браузер знав, як використовувати підключений документ. Може приймати такі значення:
  - stylesheet – визначає, що підключений файл зберігає таблицю стилів (CSS);
  - alternate – альтернативний тип використовується, наприклад, для вказання посилання на файл у форматі XML, для опису стрічки новин, анонсів статей;
- TYPE – повідомляє браузеру, який тип даних використовується для зовнішнього документа. Як правило, застосовується для того, щоб вказати, що підключений файл містить CSS;
- HREF – вказує шлях до файлу, на який посилаються URL.

Оскільки до HTML-документу підключається файл з каскадними листками стилів, то в якості значення параметра REL необхідно вказати STYLESHEET, а як значення параметра TYPE – "text/css". Оскільки це значення береться браузером за замовчуванням, параметр TYPE у тега <LINK> можна не наводити. Як значення параметра HREF необхідно вказати шлях до файла, який містить CSS.

У ***прикладі 2.1*** наведено HTML-документ, у якому, поряд з іншими способами, показано зовнішнє задання CSS у HTML-документі. У цьому прикладі як зовнішній файл, який містить CSS, використовується описаний вище файл my.css.

2.2.2. Внутрішнє задання CSS

При внутрішньому заданні каскадні листки стилів описуються так само, як і при зовнішньому, але оформляються не у вигляді зовнішнього файла, а задаються безпосередньо в області заголовка HTML-документа у середині тега-контейнера <STYLE>. . . </STYLE>.

У ***прикладі 2.1*** показано внутрішнє задання трьох каскадних аркушів стилів:

Перший CSS описує клас зеленого підкресленого тексту для тега P ("P.green_decor").

Другий CSS описує клас з вертикальним розташуванням тексту без вказівки тега (".Vert"), який, отже, можна використовувати для будь-яких тегів. Його, наприклад, можна використовувати у тег-контейнері <SPAN>…</SPAN>, який застосовується тоді, коли необхідно вказати незначні властивості не для всього абзацу тексту, а для деякого його фрагмента, аж до одного символу. У цьому випадку цей фрагмент тексту вказується у середині контейнера <SPAN>…</SPAN>.

Якщо як CSS-ідентифікатор заданий HTML-код з класом чи тільки клас, то в тегу, який використовує даний клас, необхідно вказувати параметр CLASS, який задає ім'я класу, наприклад:

```
<P CLASS="green_decor">
```

Зазначимо, що для тега зі зазначенням класу успадковуються всі властивості тега, крім тих, що описані у властивостях класу. У наведеному вище прикладі успадковується така властивість параграфа, як відступ, але змінюються такі властивості, як колір, обсяг шрифту та декоративне виділення тексту (за замовчуванням текст не виділяється).

Третій CSS описує текст, який має збільшену на 2.5 мм відстань між літерами. Оскільки для цього листка стилів як CSS-ідентифікатора заданий ідентифікатор тега ("#razr"), цей CSS може бути використаний у будь-якому тегу, в якому зазначений параметр ID з указаним значенням, наприклад:

```
<P ID="razr">
```

2.2.3. Вбудоване задання CSS

При вбудованому заданні властивості каскадних листків стилів вказуються прямо в тегу після параметра STYLE. Єдина відмінність у

синтаксисі опису CSS полягає в тому, що властивості CSS та їх значення беруться не у фігурні дужки, а в подвійні лапки.

У ***прикладі 2.1*** наведено HTML-документ, у якому показано три способи задання CSS у HTML-документі: зовнішнього, внутрішнього і вбудованого.

Приклад 2.1

```
<HTML>
<HEAD>
<TITLE> Использование CSS в HTML-документе </TITLE>
<LINK REL="stylesheet" TYPE="text/css" HREF="my.css">
<STYLE>
P.green_decor{color:green; font:16pt; text-decoration:underline}
.vert{color:#006060; font:large; writing-mode:tb-rl}
#razr{letter-spacing:2.5mm}
</STYLE>
</HEAD>
<BODY>
<H1> Параметры заголовка заданы внешним CSS </H1>
<P CLASS="green_decor"> Параметры этого параграфа заданы с помощью
внутреннего использования CSS (класс "green_decor" тэга P).
<P> Свойства абзаца, цвета и размера шрифта этого параграфа наследуются из
внешнего CSS.
<CENTER>
<SPAN CLASS="vert "> Вертикальный текст: класс vert. </SPAN>
</CENTER>
<P ID="razr">
<SPAN STYLE="color: rgb(255,0,0); font-size:1cm "> В </SPAN> строенное и
внутреннее задание CSS.
</BODY>
</HTML>
```

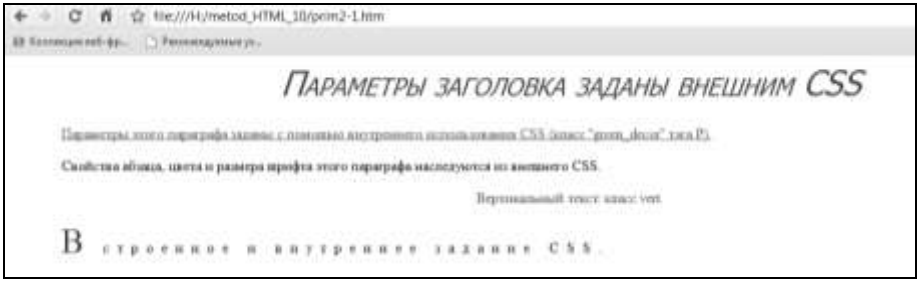


Рисунок 2.1 – Результат прикладу 2.1 на екрані

Індивідуальні завдання

Довільний текстовий документ (реферат, повідомлення, оголошення та ін.) подати за допомогою HTML і CSS у вигляді Web-сторінки, яка повинна містити:

- заголовок;
- текст, розбитий на параграфи;
- окремі фрагменти тексту (речення, слова чи символи) повинні бути згідно з таблицею 2.1 задані:
  - з допомогою зовнішнього задання CSS;
  - з допомогою внутрішнього задання CSS;
  - з допомогою вбудованого задання CSS;
  - з використанням класів (не менше трьох).

У таблиці 2.1 для кожного варіанта і для кожного виду задання CSS наведені значення властивостей CSS або назви властивостей CSS. В останньому випадку значення для зазначеної властивості вибирається студентом довільно.

Таблиця 2.1 – Варіанти індивідуальних завдань

№	Властивості CSS		
	Зовнішнє задання CSS	**Внутрішнє задання CSS**	**Вбудоване задання CSS**
1	margin-left, word-spacing	color, italic, font-family	font-size, overline
2	small-caps, color	margin-left, center, bold	word-spacing, italic
3	justify, bold	letter-spacing, underline, color	font-family, right
4	margin-left, center	font-family, bold, color	word-spacing, line-through
5	justify, color	margin-left, letter-spacing, font-size	small-caps, italic
6	margin-left, letter-spacing	color, bold, font-family	font-size, underline
7	line-through, font-size	margin-left, center, bold	word-spacing, color
8	justify, overline	word-spacing, smallcaps, color	font-family, right
9	margin-left, right	fond-size, italic, color	word-spacing, underline
10	right, color	margin-left, line-through, font-size	small-caps, bold
11	bold, letter-spacing	justify, color, right	font-family, underline
12	center, font-family	margin-left, color, line-through	word-spacing, bold
13	color, letter-spacing	justify, margin-left, italic	small-caps, font-size
14	letter-spacing, color	margin-left, font-family, underline	font-size, bold
15	font-size, margin-left	line-through, bold, color	word-spacing, center
16	overline, word-spacing	justify, small-caps, right	font-family, color
17	right, fond-size	margin-left, color, underline	word-spacing, italic
18	color, margin-left	right, font-size, bold	small-caps, writing-mode
19	right, color	margin-left, line-through, font-size	small-caps, bold
20	bold, letter-spacing	justify, color, right	font-family, underline

ЛАБОРАТОРНА РОБОТА 3
РОЗМІЩЕННЯ СПИСКІВ НА WEB-СТОРІНЦІ

Мета: вивчення засобів і можливостей HTML і CSS для подання на Web-сторінці списків із заданими параметрами.

3.1. Задання списків на Web-сторінці засобами HTML

У текстових документах використовуються списки (перерахування елементів) різних типів. HTML підтримує списки трьох видів:

- нумеровані списки;
- ненумеровані списки;
- списки-визначення.

3.1.1. Нумеровані списки

Для задання нумерованих списків, тобто таких списків, перед перерахованими елементами яких вказуються або цифри, або літери, у HTML використовується тег-контейнер <OL>. . . </OL> (OL – Ordered List, упорядкований список). Він містить два параметри:

- TYPE – задає тип нумерації:
  - 1 – арабськими цифрами (задається за умовчанням);
  - A – прописними латинськими літерами;
  - a – рядковими латинськими літерами;
  - I – прописними римськими цифрами;
  - i – рядковими римськими цифрами;
- START – задає початок нумерації списку (за замовчуванням – з першого елемента).

Самі елементи списку вказуються за допомогою тега-контейнера <LI>…</LI> (LI – List Item, елемент списку). Тег </LI>, що закриває можна не наводити.

У ***прикладі 3.1*** показано задання вкладених списків засобами HTML (у їх числі й нумеровані списки).

3.1.2. Ненумеровані списки

Для задання ненумерованих списків, тобто таких списків, перед кожним перерахованим елементом яких вказується один і той же знак, у HTML використовується тег-контейнер <UL>. . . </UL> (UL - Unordered List, невпорядкований список). Він містить параметр TYPE, значеннями якого є:

- DISC – диск (задається за умовчанням);
- CIRCLE – коло;
- SQUARE – квадрат.

Елементи ненумерованих списків так само, як і елементи нумерованих списків, задаються за допомогою тега-контейнера <LI>. . . </LI> (див. пункт 3.1.1).

Для створення вкладеного списку необхідно на місце будь-якого з елементів списку (тега <LI>) помістити тег-контейнер <UL>. . . </UL> або будь-який інший елемент, що задає список.

У ***прикладі 3.1*** показано задання вкладених списків засобами HTML (у їхньому числі і ненумеровані списки).

3.1.3. Списки-визначення

Для задання списків-визначень в HTML використовується тег-контейнер <DL>. . . </DL> (DL – Defination List, список визначень). Для задання визначень (наприклад, фрукти) використовується тег-контейнер <DT>. . . </DT> (DT – Defination Type, тип визначення). Елементи визначення перераховуються за допомогою тега-контейнера <DD>. . . </DD> (DD − Defination Data, що визначаються дані). Тег </DD>, що закриває можна не наводити.

У ***прикладі 3.1*** показано задання вкладених списків засобами HTML (у їхньому числі і списки-визначення).

Приклад 3.1

```
<HTML>
<HEAD>
<TITLE> Использование списков на Web-странице </TITLE>
<LINK REL= STYLESHEET HREF="my.css">
<STYLE>
DT{color:#000070; font:italic 0.7cm}
DD{color:#A00000; font:italic 0.6cm}
```

```
UL, OL{color:#A00000; font:italic 7.5mm}
</STYLE>
</HEAD>
<BODY>
<H1> Списки на Web-странице </H1>
<DL>
<DT> Списки, которые поддерживает HTML:
<DD> Ненумерованные с такими маркерами перечисления:
<UL TYPE=square>
<LI> disc;
<LI> square;
<LI> circle.
</UL>
<DD> Нумерованные с такими видами нумерации:
<OL START=10 TYPE=a>
<LI> арабскими цифрами;
<LI> прописными латинскими буквами;
<LI> строчными латинскими буквами;
<LI> прописными римскими цифрами;
<LI> строчными римскими цифрами.
</OL>
<DD> Списки-определения.
</DT>
</DL>
</BODY>
</HTML>
```

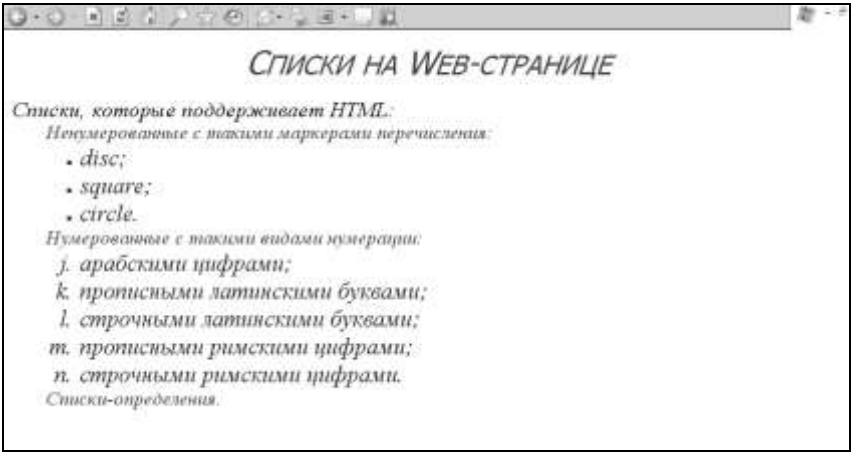


Рисунок 3.1 – Результат прикладу 3.1 на екрані

Колір, обсяг і тип шрифтів у списках у прикладі 3.1 задані за допомогою внутрішнього задання CSS для тегів DT, DL, UL і OL.

3.2. Використання CSS для задання параметрів списків

Каскадні листки стилів CSS так само, як при форматуванні тексту, використовуються для розширення можливості HTML та при заданні параметрів списків.

Для задання параметрів списків у CSS є такі властивості:

- list-style-type – змінює вигляд маркера для кожного елемента списку. Ця властивість використовується тільки у випадку, коли значення list-style-image встановлено як none. Маркери розрізняються для маркованого (ненумерованого) списку (тег <: UL>) і нумерованого (тег <: OL>).

Значення залежать від того, до якого типу списку вони застосовуються: маркірованого або нумерованого.

Маркірований список:

- disc – маркер у вигляді диска (значення за умовчанням);
- circle – маркер у вигляді кола;
- square – маркер у вигляді квадрата;

Нумерований список:

- decimal – арабські цифри (1, 2, 3, 4 ,...) (значення за умовчанням);
- decimal-leading-zero – арабські числа з нулем попереду для цифр менше десяти (01, 02, 03 ,...);
- lower-roman – римські числа в нижньому регістрі (i, ii, iii, iv, v ,...);
- upper-roman – римські числа в верхньому регістрі (I, II, III, IV, V ,...);
- lower-alpha – рядкові латинські букви a, b, c, d ,...);
- lower-latin – це значення аналогічно lower-alpha;
- upper-alpha – прописні латинські букви A, B, C, D ,...);
- upper-latin – це значення аналогічно upper-alpha;
- lower-greek – рядкові грецькі літери;
- georgian – традиційна грузинська нумерація;

– armenian – традиційна вірменська нумерація;

– none – скасовує маркери для списку;

– inherit – успадковує значення батька.

- list-style-image – задає графічне зображення маркера елементів списку, має такі значення:

– none – стиль маркера береться з властивості list-style-type (значення за умовчанням);

– url (адреса) – задає URL-адресу файл графічного зображення, якщо заданий перекриває установку list-style-type;

При заданні URL-адреси стандарт CSS не вимагає, щоб його значення бралося в лапки, але допускає використання як одиночних, так і подвійних лапок. Крім того, якщо URL-адреса сама містить символи дужок, одиночних і подвійних лапок, необхідно в адресі перед цими символами поставити символ зворотного слеша ("\").

- list-style-position – визначає, як буде розміщуватися маркер щодо тексту:

– outside – текст вирівнюється по лівому краю, а маркери розміщуються за межами текстового блоку;

– inside – маркер є частиною текстового блоку і відображається в елементі списку.

У ***прикладі 3.2*** показано використання властивостей list-style-type і list-style-image.

Приклад 3.2

```
<HTML>
<HEAD>
<TITLE> Создание списков с помощью CSS </TITLE>
<LINK REL= STYLESHEET HREF="my.css">
<STYLE>
.image{color:#000070; font:italic 0.5cm; list-style-
image: url(tre1.gif)}
</STYLE>
</HEAD>
<BODY>
<H1> Списки на Web-странице </H1>
<P> CSS позволяет формировать на Web-странице списки:
```

```
<UL CLASS="image" STYLE="list-style-position:inside">
<LI> с заданием знаков перечесления с помощью свойства
list-style-type:
<UL style="color:#000070; font:italic 0.5cm;
list-style-type:decimal; list-style-image:none">
<LI> disc;
<LI> square;
<LI> circle;
<LI> decimal;
<LI> lower-roman;
<LI> upper-roman;
<LI> lower-alpha;
<LI> upper-alpha.
</UL>
<LI> с заданием знаков перечисления в виде произвольных
изображений ( ) с помощью
свойства list-style-image.
</UL>
</BODY>
</HTML>
```

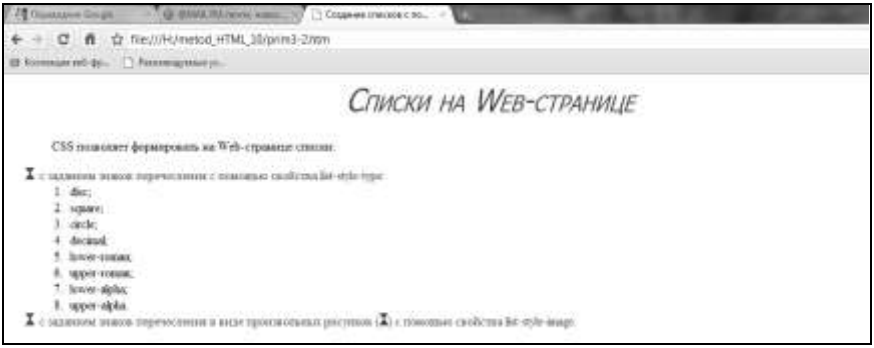


Рисунок 3.2 – Результат прикладу 3.2 на екрані

Колір, обсяг і тип шрифтів у списках у **прикладі 3.2** задані для тега UL зовнішнього списку за допомогою внутрішнього задання CSS, для тега UL внутрішнього списку – за допомогою вбудованого.

Індивідуальні завдання

Сформувати на Web-сторінці два вкладених списки:

- перший складається зі списку № 1 (зовнішній) і списку № 2 (внутрішній);
- другий складається зі списку № 3 (зовнішній) і списку № 4 (внутрішній);
- значення параметрів для списків № 1, № 2, № 3 та № 4 для кожного варіанта наведені у таблиці 3.1;
- кожний із чотирьох списків повинен містити не менше п'яти елементів.

Таблиця 3.1 – Варіанти індивідуальних завдань

№	Список № 1	Список № 2	Список № 3	Список № 4
1	2	3	4	5
1	disc(HTML)	decimal	A/10	url()
2	url()	circle(HTML)	lower-roman	a/7
3	I/101	url()	square(HTML)	upper-roman
4	lower-alpha	i/77	url()	disc(HTML)
5	upper-alpha	circle(HTML)	1/25	url()
6	url()	disc(CSS)	square(HTML)	A/21
7	a/11	url()	circle(CSS)	disc(HTML)
8	circle(HTML)	I/5	url()	square (CSS)
9	square(HTML)	decimal	url()	i/20
10	1/13	disc(HTML)	lower-roman	url()
11	url()	A/8	circle(HTML)	upper-roman
12	lower-alpha	url()	1/17	square(HTML)
13	disc(HTML)	A/22	upper-alpha	url()
14	url()	circle(HTML)	a/3	disc(CSS)

Продовження табл. 3.1

1	2	3	4	5
15	circle(CSS)	url()	disc(HTML)	i/1991
16	i/255	square(CSS)	url()	circle(HTML)
17	1/25	decimal	square(HTML)	url()
18	url()	A/10	lower-roman	disc(HTML)
19	circle(HTML)	url()	a/16	upper-roman
20	lower-alpha	square (HTML)	url()	i/125

Примітка: для нумерованих списків параметри задані у вигляді TYPE/START.

ЛАБОРАТОРНА РОБОТА 4
РОЗМІЩЕННЯ ТАБЛИЦЬ НА WEB-СТОРІНЦІ

Мета: вивчення засобів і можливостей HTML і CSS для подання на Web-сторінці таблиць із заданими параметрами.

4.1. Подання таблиць на Web-сторінці

HTML має великі можливості для подання на Web-сторінці різних таблиць. Необхідно зазначити, що засоби формування таблиць використовуються в HTML не тільки для зображення власне таблиць, а і для розміщення елементів Web-сторінки на екрані. У цьому випадку екран браузера подається у вигляді таблиці з невидимими межами, і окремі елементи Web-сторінки поміщаються в ту чи іншу комірку таблиці, причому окремі комірки самі можуть бути оформлені у вигляді таблиць. Недоліком такого підходу є те, що при великому обсязі таблиці завантажуватися Web-сторінка буде довго.

Іншим більш сучасним способом верстки Web-сторінок є використання шарів.

4.1.1. Задання параметрів таблиці

Таблиця на мові HTML вказується за допомогою тега-контейнера <TABLE>. . . </TABLE> для елементів, що визначають зміст таблиці. Будь-яка таблиця складається з рядків і комірок, які задаються відповідно за допомогою тегів <TR> і <TD>. Всередині <TABLE>. . . </TABLE> допустимо використовувати такі елементи: <CAPTION>, <COL>, <COLGROUP>, <TBODY>, <TD>, <TFOOT>, <TH>, <THEAD> і <TR>. Тег </TABLE>, що закриває, є обов'язковим. Загальні параметри таблиці задаються у тегу <TABLE>, який містить такі параметри:

- ALIGN – вирівнювання таблиці (аналог CSS - text-align):
  - left – по лівому краю (значення за умовчанням);
  - center – по центру;
  - right – по правому краю;
- WIDTH – задає ширину таблиці (аналог CSS − width). Якщо загальна ширина змісту перевищує зазначену ширину таблиці, то браузер буде намагатися "втиснутися" в задані розміри за рахунок форматування тексту. У випадку, коли це неможливо, наприклад, таблиця містить

зображення, параметр width буде проігноровано, і нова ширина таблиці буде обчислена на основі її змісту.

Приймає будь-яке ціле значення в пікселях або відсотках від доступного простору.

- HEIGHT – встановлює висоту таблиці (аналог CSS - height). У специфікації HTML 4 цього параметра немає, проте браузери в більшості випадків розуміють його, якщо не встановлено тег прологу <!DOCTYPE>. Коли в документі заданий цей тег, браузери висоту таблиці, яка задана через параметр HEIGHT, ігнорують.

Приймає будь-яке ціле значення в пікселах або відсотках від доступного простору.

- COLS – задає кількість стовпців у таблиці, допомагаючи браузеру в підготовці до її відображення. Без цього параметра таблиця буде показана тільки після того, як весь зміст таблиці буде додано у браузер і проаналізовано. Використання параметра COLS дозволяє трохи прискорити подання вмісту таблиці.

- BORDER – встановлює товщину рамки в пікселях (аналог CSS - border-width). За замовчуванням рамка зображується тривимірною, але якщо використовується параметр BORDERCOLOR тега <TABLE>, то вид рамки змінюється в залежності від браузера. Коли в тегу <TABLE> використовується параметр BORDER без значень (<TABLEBORDER>), то браузер відображає рамку товщиною один піксель. Якщо параметр BORDER не заданий, то рамка взагалі не відображається.

- BORDERCOLOR – встановлює колір рамки таблиці (аналог CSS - background-color). Рамка зазвичай рисується як тривимірна, додавання параметрів BORDERCOLOR і BORDER до тегом <TABLE> створюють однотонну лінію.

- BGCOLOR – встановлює колір фону таблиці. Можна використовувати цей параметр спільно з BACKGROUND, підібравши колір фону близький до фонового рисунку (аналог CSS - background-color).

- BACKGROUND – визначає зображення, яке буде використовуватися в якості фонового рисунка таблиці (аналог CSS - background-image). На відміну від звичайних зображень, для фону не встановлюються ширина і висота, і він завжди відображається в натуральну величину з масштабом 100%. Якщо рисунок за розміром менше ширини або висоти таблиці, то картинка повторюється по горизонталі вправо і вниз, вибудовується, як мозаїка. При виборі фону потрібно переконатися, що був

забезпечений достатній контраст між ним і змістом таблиці. Як фон допускається використовування анімованих зображень у форматі GIF, але вони відволікають увагу читачів.

Значенням є будь-який допустимий адрес зображення – можна використовувати відносний або абсолютний шлях.

- CELLSPACING – задає відстань у пікселях або у відсотках вільного простору між зовнішніми межами елементів таблиці (обмежувальні рамки включаються в комірку), за замовчуванням дорівнює 0, якщо не заданий параметр BORDER і 2, якщо заданий.
- CELLPADDING − визначає відстань у пікселях або у відсотках вільного простору між межею комірки і її змістом. Цей параметр додає порожній простір до комірки, збільшуючи тим самим її розміри. Без CELLPADDING текст в таблиці примикає до рамки, знижуючи тим самим його сприйняття. Додавання ж CELLPADDING дозволяє поліпшити читабельність тексту. При відсутності меж особливого значення цей параметр не має, але може допомогти, коли потрібно встановити порожню відстань між комірками. За замовчуванням дорівнює 1 (аналог CSS − padding).
- FRAME − повідомляє браузеру, як відображати межі навколо таблиці (аналог CSS - border):
  - border – рисуються всі лінії зовнішньої рамки (значення за умовчанням).
  - above – рисується тільки верхня лінія зовнішньої рамки;
  - below – рисується тільки нижня лінія зовнішньої рамки;
  - hsides – рисуються тільки горизонтальні лінії зовнішньої рамки, тобто верхня і нижня;
  - vsides – рисуються тільки вертикальні лінії зовнішньої рамки, тобто ліва і права;
  - lhs – рисується тільки ліва лінія зовнішньої рамки;
  - rhs – рисується тільки права лінія зовнішньої рамки;
  - void – зовнішньої рамки немає взагалі.
- RULES – повідомляє браузеру, де відображати кордону між осередками (аналог CSS - border). Товщина кордону і її колір вказується за допомогою параметрів BORDER і BORDERCOLOR. За замовчуванням рамка рисується навколо кожного осередку, утворюючи тим самим сітку. Може приймати наступні значення:

– all – рисуються всі внутрішні лінії (значення за умовчанням, якщо значення параметра BORDER відмінно від нуля);

– rows – рисуються тільки горизонтальні лінії (між рядками, заданими тегом <ROW>);

– cols – рисуються тільки вертикальні лінії (між стовпцями);

– none – не рисуються ніякі внутрішні лінії (значення за умовчанням, якщо значення параметра BORDER дорівнює нулю);

– groups – лінія відображається між групами, які утворюються тегами <THEAD>, <TFOOT>, <TBODY>, <COLGROUP> або <COL>.

Заголовок таблиці, якщо він необхідний, задається за допомогою тега-контейнера <CAPTION>. . . </CAPTION>, який може розміщуватися тільки відразу після відкриваючого тега <TABLE>. Такий заголовок являє собою текст, за замовчуванням відображається перед таблицею і описує її зміст. Може мати два пареметра:

- ALIGN – визначає вирівнювання заголовка стосовно таблиці. Результат його дії залежить від використовуваного браузера і встановленого значення (аналог CSS - text-align):

– left – в браузері Internet Explorer і Opera розпорядженні заголовок зверху і вирівнює його по лівому краю таблиці. У Firefox заголовок розташовується зліва від таблиці. Safari і Chrome ігнорують це значення;

– center – по центру над таблицею (значення за умовчанням);

– right – в браузері Internet Explorer і Opera розміщує заголовок зверху таблиці і вирівнює його по правому краю таблиці. У браузері Firefox заголовок розташовується від таблиці справа. Safari і Chrome ігнорують це значення;

– top – результат аналогічний дії значення center, але на відміну від нього входить в специфікацію HTML 4 і розуміється усіма браузерами (значення за умовчанням);

– bottom – знизу по центру таблиці.

- VALIGN – призначений для розміщення заголовка до таблиці або після неї. Цей параметр працює тільки в браузері Internet Explorer, а для всіх інших застосовується ALIGN. Параметр ALIGN коректно діє і для IE, тому поєднання параметрів ALIGN і VALIGN необхідно тільки для цього браузера, коли потрібно одночасне вирівнювання заголовка по вертикалі і горизонталі. Приймає такі значення:

– top – заголовок розміщується над таблицею;

– bottom – заголовок розміщується внизу таблиці.

Використання параметра VALIGN не рекомендується специфікацією HTML.

У ***прикладі 4.1*** показано задання параметрів таблиці.

4.1.2. Формування таблиці

Після задання параметрів і заголовка таблиці виконується формування її структури та змісту.

Побудова таблиці здійснюється порядково, кожен рядок задається за допомогою тега-контейнера <TR> (TR – Table Row, рядок таблиці). . . </TR> (тег, що закриває, необов'язковий), який має наступні параметри:

BORDERCOLOR і BGCOLOR мають ті ж призначення, що і відповідні параметри тега <TABLE> (див. пункт 4.1.1), але стосовно до параметрів рядка таблиці. Вирівнювання здійснюється для всіх комірок у межах одного рядка. Якщо потрібно застосувати різне вирівнювання для кожної комірки, можна скористатися стилями або використовувати параметр ALIGN для тега <TD> або <TH> (аналог CSS - text-align):

- ALIGN – задає горизонтальне вирівнювання змісту в комірках рядка:
  - left – по лівому краю комірки (значення за умовчанням);
  - center – по центру комірки;
  - right – по правому краю комірки;
  - justify – по всій ширині комірки;
- VALIGN – ставить вертикальне вирівнювання тексту у комірках рядка (аналог CSS – vertical-align):
  - top – по верхньому краю комірки;
  - middle – по центру комірки (значення за умовчанням);
  - bottom – по нижньому краю комірки;
  - baseline − за базовою лінією комірки, при цьому відбувається прив'язка змісту комірки до однієї лінії.

Формування кожного рядка виконується по коміркам рядка зліва направо за допомогою тегів-контейнерів <TH>. . . </TH> (TH – Table Head, заголовок таблиці), які задають назви (заголовки) стовпців таблиці, і тегів-контейнерів <TD>. .. </TD> (TD – Table Data, дані таблиці), які задають значення елементів таблиці.

Обидва теги <TH> і <TD> задають параметри і зміст комірки. Різниця між ними полягає в тому, що <TH> за умовчанням містить напівжирний текст і вирівняний по центру, а <TD> – містить звичайний текст і вирівняний по лівому краю.

Теги <TD> і <TH> мають такі параметри:

- ALIGN – визначає вирівнювання змісту комірки по горизонталі, має ті ж значення, що і для тега <TR>;
- VALIGN – визначає вирівнювання змісту комірки по вертикалі, має ті ж значення, що і для тега <TR> (аналог CSS - vertical-align);
- WIDTH – задає ширину комірки, має ті ж значення, що і для тега <TABLE>;
- HEIGHT – задає висоту комірки, має ті ж значення, що і для тега <TABLE>;
- BORDERCOLOR – задає колір рамки;
- BGCOLOR – задає колір фону комірки;
- BACKGROUND – задає фоновий рисунок комірки;
- NOWRAP – додавання цього параметра до тегу <TD> змушує текст всередині комірки відображатися без переносів, одним суцільним рядком. Неправильне використання цього параметра може призвести до того, що таблиця буде занадто широкою і не поміститься цілком на веб-сторінку. Як наслідок, з'явиться горизонтальна смуга прокручування. Цей параметр не рекомендований до використання в специфікації HTML 4 (аналог CSS - white-space);
- COLSPAN – вказує число комірок, об'єднаних по горизонталі в одну комірку;
- ROWSPAN – вказує число комірок, об'єднаних по вертикалі в одну комірку.

У ***прикладі 4.1*** показано побудову таблиці з заданням параметрів тегів <TABLE>, <TR>, <TH> і <TD>.

4.1.3 Логічне форматування таблиці

При логічному форматуванні таблиці її рядки розбиваються на групи, що представляють заголовок таблиці, її тіло і основу. Для цього використовуються теги <THEAD>, <TBODY>, <TFOOT>, <COLGROUP> і <COL>, що задають логічну структуру таблиці:

- <THEAD> – призначений для подання одного або кількох рядків, які представлені вгорі таблиці. Допустимо використовувати не більше одного елемента <THEAD> в межах однієї таблиці, і він повинен йти у вихідному коді відразу після тега <TABLE>;
- <TBODY> – призначений для подання одного або кількох рядків таблиці. Це дозволяє створювати структурні блоки, до яких можна застосовувати єдине оформлення через стилі, а також керувати їх видом через скрипти. Допускається застосовувати декілька тегів <TBODY> всередині контейнера <TABLE>. Не повинен перекриватися з іншими видами угруповань рядків – <TFOOT> і <THEAD>;
- <TFOOT> – призначений для подання одного або кількох рядків, які представлені внизу таблиці. Хоча тег <TFOOT> у вихідному коді має бути визначений до тега <TBODY>, браузери відображають його в самому низу таблиці. У межах таблиці дозволяється використовувати тільки один елемент <TFOOT>;
- <COLGROUP> – призначений для задання ширини і стилю однієї або декількох колонок таблиці. Цей тег дозволяє зменшити код таблиці за рахунок скорочення повторюваних параметрів, і при наявності цього тега браузер починає показувати вміст таблиці, не чекаючи її повного завантаження. Тег <COLGROUP> можна використовувати в комбінації з міткою <COL>, який визначає характеристики однієї або декількох колонок. Зазвичай закривать тег не потрібно, але якщо <COLGROUP> виступає як контейнер для елементів <COL>, тоді слід додати тег </COLGROUP> у кінці групи.

Різниця між властивостями тегів <COLGROUP> і <COL> не дуже велика і полягає в наступному. Тег <COLGROUP> дозволяє об'єднувати колонки в певні групи, також при додаванні параметра RULES = "groups" до тегу <TABLE> браузер буде рисувати лінію тільки між колонками, створеними за допомогою <COLGROUP>. В інших випадках поведінка колонок, призначених через елементи <COLGROUP> і <COL>, ідентично.

- <COL> – задає ширину і інші характеристики однієї чи кількох колонок таблиці. При наявності цього тега браузер починає показувати зміст таблиці, не чекаючи її повного завантаження. Тег <COL> можна використовувати спільно з тегом <COLGROUP>, який задає групу колонок, що володіють загальними параметрами.

4.1.4. Використання CSS для задання параметрів таблиць

При побудові таблиці можуть бути задіяні розширені можливості CSS.

Властивість width встановлює ширину блокових або замінних елементів (до них, наприклад, відноситься тег <IMG>). Ширина не включає товщину меж навколо елемента, значення відступів і полів.

Браузери неоднаково працюють з шириною, результат відображення залежить від використовуваного тега DOCTYPE. У таблиці 4.1 наведені можливі варіанти DOCTYPE і одержувана ширина.

Таблиця 4.1 – Дія властивості width в браузерах Internet Explorer, Firefox і Opera

<table>
<tr><th>DOCTYPE</th><th>Internet Explorer</th><th>Firefox</th><th>Opera</th></tr>
<tr><td>Не указан</td><td rowspan="2">Якщо вміст перевищує задану ширину, то блок змінює свої розміри, підлаштовуючись під вміст. В іншому випадку ширина блоку дорівнює значенню width.</td><td rowspan="2">У всіх випадках Firefox діє по специфікації CSS. А саме, ширина блоку виходить складанням значень width, padding, margin і border.</td><td>Ширина дорівнює значенню width.</td></tr>
<tr><td>"Перехідний" <DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"></td><td>Зміст блоку, якщо не поміщається в задані розміри, відображається поверх блоку.</td></tr>
<tr><td>"Суворий" <l;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"></td><td>Ширина формується шляхом складання значень width, padding, margin і border.</td><td>Зміст блоку, якщо не поміщається в задані розміри, відображається поверх блоку.</td><td>Ширина дорівнює значенню width плюс padding, margin і border.</td></tr>
</table>

Значення властивості width можуть задаватися:

- в одиницях довжини, допустимих в CSS;
- у відсотках від ширини батьківського елементів (якщо батько явно не вказаний, то в його якості виступає вікно браузера);
- auto – встановлює ширину, враховуючи тип і зміст елемента (задається за замовчуванням);
- inherit – успадковує значення батька.

Властивість height встановлює ширину блокових або замінних елементів (до них, наприклад, відноситься тег <IMG>). Висота не включає товщину меж навколо елемента, значення відступів і полів.

Якщо зміст блоку перевищує зазначену висоту, то висота елемента залишиться незмінною, а зміст відображатиметься поверх нього. Через цю особливість може вийти накладення змісту елементів один на одного, коли елементи в коді HTML йдуть послідовно. Щоб цього не сталося, необхідно додати overflow: auto до стилю елемента.

Значення властивості <= "" span = ""> можуть задаватися:

- в одиницях довжини, допустимих в CSS;
- у відсотках від висоти батьківського елемента (якщо батько явно не вказаний, то в його якості виступає вікно браузера);
- auto – встановлює висоту, враховуючи тип і зміст елемента (задається за замовчуванням);
- inherit - успадковує значення батька.

При заданні меж елементів Web-сторінки можна використовувати такі властивості CSS:

- border-color – вказує колір межі, може бути задано:
  - ransparent – задає прозорий колір;
  - inherit – успадковує значення батька,

  або від одного до чотирьох значень:

  – якщо задано одне значення – воно застосовується до всіх чотирьох меж;

  – якщо задано два значення – перше належить до верхньої і нижньої меж, друге – до лівої і правої меж;

  – якщо задано три значення – перше належить до верхньої межі,

  – друге – до лівої і правої меж, третє – до нижньої межі.

  – якщо задано чотири значення – перше належить до верхньої межі, друге – до лівої межі, третє – до правої межі, четверте – до нижньої межі.
- border-style – вказує стиль межі:
  - none – меж не видно (значення приймається за умовчанням);
  - solid – суцільна лінія;
  - double – подвійна суцільна лінія;

– dotted – лінія у вигляді точок;
– dashed – пунктирна лінія;
– groove – тривимірна увігнута межа;
– ridge – тривимірна випукла межа;
– inset – тривимірна межа "сходинка вгору";
– outset – тривимірна межа "сходинка вниз";
– inherit – успадковує значення батька..

- border-width – вказує товщину межі, яка може бути задана у вигляді числового значення в пікселях або за допомогою таких значень:

– thin – тонка лінія;
– medium – лінія середньої товщини (значення прийняте за умовчанням);
– thick – товста лінія;
– inherit – успадковує значення батька.

Може приймати від одного до чотирьох значень, число яких має таке ж значення, що і для властивості border-color.

- border-collapse – встановлює, як відображати межі навколо комірок таблиці. Це властивість грає роль, коли для комірок встановлена рамка, тоді в місці стику комірок вийде лінія подвійної товщини. Значення collapse змушує браузер аналізувати подібні місця в таблиці і прибирати в ній подвійні лінії. При цьому між осередками залишається тільки одна межа, одночасно належить обом коміркам. Те ж правило дотримується і для зовнішніх меж, коли навколо самої таблиці додається рамка (застосовується тільки для тега <TABLE>). Може приймати такі значення:

– separate – поділяє межу таблиці і межі її комірок (значення прийняте за умовчанням);
– collapse – об'єднує їх;
– inherit – успадковує значення батька.

Параметри межі, як і параметри шрифту, можна вказувати у скороченій формі. Наприклад, замість задання властивостей межі у вигляді:

```
<TD STYLE="border-color:green; border-style:dotted;
border-width:6">
```

можна використовувати таку форму:

```
<TD STYLE="border:green dotted 6">
```

Якщо при цьому вказані ні всі властивості межі, то інші успадковуються або приймаються за умовчанням.

Крім задання властивостей відразу декількох меж елемента, CSS мають можливість задання властивостей тільки однієї межі:

- border-top – верхньої межі;
- border-bottom – нижньої межі;
- border-left – лівої межі;
- border-right – правої межі.

Для вказівки відразу декількох властивостей кожної з цих меж можна використовувати скорочену форму задання значень, як наведено вище. Для задання однієї з властивостей меж можна скористатися однією з таких властивостей:

- border-top-style – задання стилю верхньої межі;
- border-top-width – задання ширини верхньої межі;
- border-top-color – задання кольору верхньої межі;
- border-bottom-style – задання стилю нижньої межі;
- border-bottom-width – задання ширини нижньої межі;
- border-bottom-color – задання кольору нижньої межі;
- border-left-style – задання стилю лівої межі;
- border-left-width – задання ширини лівої межі;
- border-left-color – задання кольору лівої межі;
- border-right-style – задання стилю правої межі;
- border-right-width – задання ширини правої межі;
- border-right-color – задання кольору правої межі.

У ***прикладі 4.1*** показано використання властивостей CSS при побудові таблиці.

4.2. Задання рухомих елементів на Web-сторінці

Для задання рухомих елементів на Web-сторінці використовується тег-контейнер <MARQUEE>…</MARQUEE>. Рухомий елемент (за який може бути використаний текст, рисунок, таблиця та інші елементи Web-сторінки) розміщується всередині контейнера, тобто між тегами <MARQUEE> і </MARQUEE>. Тег <MARQUEE> має такі параметри:

- SCROLLDELAY – вказує затримку в мс (значення за замовченням 85);
- SCROLLAMOUNT – встановлює відстань у пікселях між старим і новим положеннями змісту елементу <MARQUEE> при його анімації, яка відбувається за рахунок періодичного стирання інформації та відображення її на новому місці, тим самим впливаючи на швидкість руху і плавність ходу. Чим вище значення SCROLLAMOUNT, тим швидше переміщається зміст контейнера. Якщо значення дорівнює нулю, то рух не відбувається (значення за замовчуванням 6);
- TRUESPEED – скасовує вбудований обмежувач швидкості при невеликих значеннях параметра SCROLLDELAY. Якщо зміст тега <MARQUEE> переміщається занадто швидко, то спрацьовує вбудований обмежувач швидкості, який насильно уповільнює прокрутку. Це зроблено для того, щоб зберегти читабельність тексту. Щоб обійти це обмеження, необхідно додати параметр TRUESPEED, він змусить прокручувати зміст із заданою швидкістю. Для параметра SCROLLDELAY обмеження швидкості починається зі значення 60 мілісекунд і нижче;
- LOOP – задає, скільки разів буде прокручуватися зміст. Після того, як задане число раз відрахувати, зміст залишається в кінцевій точці. Значенням є будь-яке ціле позитивне число або -1 для нескінченного відтворення руху (значення за замовчуванням дорівнює -1);
- BGCOLOR – задає колір фону (за замовчуванням збігається з кольором фону Web-сторінки);
- BEHAVIOR – вказує характер руху:
  - scroll – елемент (контент) переміщається у напрямку, заданим параметром DIRECTION, потім ховається за межами області, після чого починає рух з початку. (задається за замовчуванням);
  - alternate – елемент переміщається по черзі то в один бік, то в інший;
  - slide – елемент переміщається до краю області і зупиняється;
- DIRECTION – вказує напрямок руху елемента:
  - left – справа наліво (задається за замовчуванням);
  - right – зліва направо;
  - up – знизу вгору;

– down – зверху вниз;

- WIDTH і HEIGHT – задають ширину і висоту області прокручування. Елемент MARQUEE відображається як прямокутник, ширина і висота якого встановлюються, відповідно, за допомогою параметрів WIDTH і HEIGHT. Прокрутка змісту відбуватиметься в межах заданих розмірів. Допускається використовувати значення в пікселах або відсотках. Якщо встановлен процентний запис, то розміри <MARQUEE> обчислюються щодо батьківського елементу, звичайно це тег BODY.

Значення за замовчуванням: ширина – 100%, висота – 12 пікселів;

- HSPACE і VSPACE – задають у пікселах горизонтальні і вертикальні поля навколо контенту.

Тег <MARQUEE> не входить в специфікацію HTML і його наявність призведе до невалідного коду.

Як приклад побудови таблиці на Web-сторінці наведемо такий HTML-документ:

Приклад 4.1

```
<HTML>
<HEAD>
<TITLE> Использование таблиц на Web-странице </TITLE>
<LINK REL=STYLESHEET HREF="my.css">
<STYLE>TABLE{font: italic 6mm; color:#00A}</STYLE>
</HEAD>
<BODY>
<H1> ТАБЛИЦЫ НА WEB-СТРАНИЦЕ </H1>
<TABLE BORDER BGCOLOR=#E0E0E0 CELLSPACING=4
CELLPADDING=3 ALIGN=center WIDTH=90%>
<CAPTION ALIGN=bottom> ТАБЛИЦА 1 </CAPTION>
<TR BGCOLOR=aqua>
<TH COLSPAN=3> Заголовок столбцов 1-3 <TH> Столбец 4
<TR>
<TD ALIGN=right STYLE="border:solid 10; border-color:red
blue green yellow"> 1/1
<TD ALIGN=center STYLE="border-width:thick"> 1/2
<TD ROWSPAN=2 VALIGN=top>
<MARQUEE BEHAVIOR=slide DIRECTION=down SCROLLAMOUNT=2>
1-2/3
```

```
</MARQUEE><TD> 1/4
<TR>
<TD COLSPAN=2 BGCOLOR=#E0E0FF STYLE="font-size:8mm;
border:blue 10 dotted">
<MARQUEE BEHAVIOR=alternate SCROLLAMOUNT=10 LOOP=4> 2/1-
2 </MARQUEE>
<TD VALIGN=bottom ALIGN=center
STYLE="font-size:2cm; border-left: #F00 dashed 16 " >
2/4
</TABLE>
</BODY>
</HTML>
```

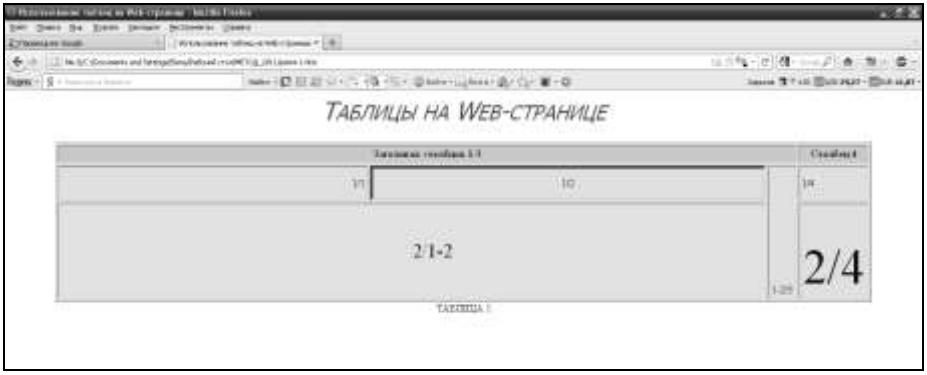


Рисунок 4.1- Результат прикладу 4.1 на екрані

Індивідуальні завдання

На Web-сторінці розробити таблицю з параметрами, вказаними у таблиці 4.1:

- розмір таблиці: заданий у вигляді MxN, де M – кількість рядків таблиці; N - кількість стовпців таблиці;
- об'єднання:
  - стовпців – задано у вигляді M (N1-N2), де M – номер рядка, N1, N2 - початковий і кінцевий номери об'єднаних стовпців;
  - рядків – задано у вигляді N (M1-M2), де N – номер стовпця, M1, M2 - початковий і кінцевий номери об'єднаних рядків;

- виділений елемент таблиці вказує тип елемента таблиці і властивості його ліній.
- перший рядок задати як заголовок стовпців.

Таблиця 4.1 – Варіанти індивідуальних завдань

№	Розмір таблиці	Об'єднання		Виділений елемент таблиці				
		стовпців	рядків	Тип	Властивості лінії			
					вид	тип	товщина	колір
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	3x7	1 (3-5)	6 (2-3)	TH	border-bottom	dashed	10	червоний
2	5x5	4 (2-4)	5 (1-5)	TD	border	dotted	7	синій
3	6x4	5 (1-2)	2 (1-5)	TR	border-left	double	5	жовтий
4	2x8	2 (3-7)	3 (1-2)	TABLE	border-right	outset	11	зелений
5	4x5	3 (4-5)	1 (1-3)	TH	border-top	inset	4	бірюзовий
6	7x4	6 (1-4)	2 (2-6)	TD	border-bottom	double	thick	ліловий
7	4x8	3 (3-7)	2 (2-4)	TR	border	dashed	12	темно-бірюзовий
8	6x6	3 (1-6)	5 (4-5)	TABLE	border-left	dotted	9	темно-червоний
9	4x7	1 (2-5)	5 (1-3)	TH	border-right	inset	6	темно-синій
10	6x3	5 (2-3)	3 (2-6)	TD	border-top	dashed	10	темно-зелений
11	8x3	7 (1-2)	3 (1-7)	TR	border-bottom	dotted	thick	темно-ліловий
12	7x5	5 (3-5)	2 (1-3)	TABLE	border	outset	medium	світло-бірюзовий
13	7x7	2 (4-7)	7 (2-5)	TH	border-left	inset	11	світло-червоний
14	3x9	3 (3-9)	8 (2-3)	TD	border-right	dotted	8	світло-синій

Продовження табл. 4.1

1	2	3	4	5	6	7	8	9
15	9x3	3 (1-3)	2 (5-9)	TR	border-top	double	3	світло-зелений
16	4x8	4 (3-6)	7 (2-4)	TABLE	border-bottom	dashed	thick	світло-бірюзовий
17	6x5	2 (3-4)	4 (2-5)	TH	border	dotted	9	червоний
18	5x7	3 (5-7)	4 (1-3)	TD	border-left	double	thick	синій
19	3x6	1 (2-5)	3 (1-3)	TR	border-right	dashed	10	зелений
20	8x8	7 (3-6)	6 (2-7)	TABLE	border-top	outset	thin	бірюзовий

ЛАБОРАТОРНА РОБОТА 5
РОЗМІЩЕННЯ ЗОБРАЖЕНЬ НА WEB-СТОРІНЦІ

Мета: вивчення засобів і можливостей HTML і CSS для подання на Web-сторінці зображень, відеофрагментів і звуку.

5.1. Подання зображень на Web-сторінці

5.1.1. Задання параметрів зображення

Для розміщення графічних зображень на Web-сторінці застосовується тег <IMG>, який використовує такі параметри:

- SRC (єдиний обов'язковий параметр) – задає URL-адресу файла з графічним зображенням. URL (Universal (Uniform) Resource Locator) має такий формат: http://host [: port] [path], де
  - http (hypertext transfer protocol) – назва протоколу передачі гіпертексту (HTML-документа Web-сторінки);
  - host – IP-адреса або доменне ім'я комп'ютера;
  - port – номер порту;
  - path – шлях до файла на комп'ютері.
- ALT – встановлює альтернативний текст для зображень. Такий текст дозволяє отримати текстову інформацію про зображення у випадку, коли неможливо зображення відтворити на екрані, наприклад, при відключені в браузері завантажень зображень або незнаходження файла з зображенням. Оскільки завантаження зображень відбувається після отримання браузером інформації про нього, то текст з'являється раніше зображення. Потім після завантаження текст змінюється зображенням.

Значенням є будь-який відповідний текстовий рядок. Його обов'язково треба брати в подвійні або одинарні лапки.

Браузери також відображають альтернативний текст у вигляді підказки, що з'являється при наведенні курсору миші на зображення (якщо поряд з параметром ALT використовується також параметр TITLE, то підказка буде задана саме цим параметром);

- ALIGN – вирівнювання зображення щодо тексту:
  - left – зображення розташовується зліва від тексту;
  - right – зображення розташовується праворуч від тексту;

– top – верхній край зображення вирівнюється по верхньому краю тексту;

– middle – центр зображення вирівнюється по нижньому рядку тексту;

– bottom – нижній край зображення вирівнюється за базовою лінією тексту (задається за умовчанням).

- BORDER – задає товщину рамки навколо зображення у пікселях (аналог CSS – властивість border). За замовчуванням рамка навколо зображення не відображається за винятком випадку, коли рисунок є посиланням. При цьому колір рамки співпадає з кольором посилань, заданих за допомогою стилю або параметра LINK тега <BODY>. Щоб прибрати рамку, слід задати параметр BORDER = "0".
- HEIGHT – висота зображення (аналог CSS - властивість height);
- WIDTH – ширина зображення (аналог CSS - властивість width).

Для параметрів WIDTH і HEIGHT допускається використовувати значення в пікселях або відсотках. Якщо встановлений процентний запис, то розміри зображення обчислюються щодо батьківського елементу – контейнера, де знаходиться тег <IMG>. У разі відсутності батьківського контейнера, в його якості виступає вікно браузера.

- HSPACE – задає відступ в пікселях від навколишнього контента по горизонталі;
- VSPACE – задає відступ в пікселях від навколишнього контента по вертикалі;
- ISMAP – показує браузеру, що рисунок є картою-зображенням;
- USEMAP – встановлює зв'язок з тегом <MAP>, що містить координати для карти-зображення, при цьому до якості значення параметра USEMAP необхідно вказати значення параметра NAME тега <MAP>, додавши попереду символ решітки ("#").

Вказівка тільки одного параметра WIDTH або HEIGHT зберігає пропорції і відношення сторін зображення. Браузер при цьому очікує повного завантаження рисунка, щоб визначити його початкову висоту і ширину.

Обов'язково задавайте розміри всіх зображень на Web-сторінці. Це дещо прискорює завантаження сторінки, оскільки браузеру немає потреби обчислювати розмір кожного рисунка після його отримання. Це твердження особливо важливо для зображень, розміщених всередині таблиці.

Ширину і висоту зображення можна змінювати як в меншу, так і велику сторону. Проте на швидкість завантаження рисунка це ніяк не впливає, оскільки розмір файла залишається незмінним. Тому з обережністю зменшуйте зображення, т.к. це може викликати здивування у читачів, чому такий маленький рисунок так довго завантажується. А ось збільшення розмірів призводить до зворотного ефекту - розмір зображення великий, але файл щодо зображення аналогічного розміру завантажується швидше. Але якість зображення при цьому погіршується.

Обсяг елементів на Web-сторінці можна також змінити за допомогою властивості CSS zoom, яка задає коефіцієнт масштабування для збільшення або зменшення обсягу. Значення можуть бути задані у вигляді:

- числа з плаваючою точкою;
- процентного запису;
- зумовленого значення auto (приймається за умовчанням і дорівнює 1.0 або 100%).

Для будь-якого зображення можна задати відступи по горизонталі та вертикалі за допомогою параметрів HSPACE і VSPACE. Особливо це актуально при обтіканні зображення текстом. У цьому випадку, щоб текст не "наїжджав" щільно на зображення, необхідно навколо нього додати порожній простір.

У ***прикладі 5.1*** показано задання параметрів зображення при розміщенні його на екран (файл "fish.gif"). Використання параметрів ISMAP і USEMAP показано в лабораторній роботі 6 ("Робота з посиланнями на Web-сторінці", ***приклад 6.1***).

5.1.2. Задання фону у вигляді зображення

Для задання фону у вигляді зображення у тегах, в яких можна вказувати фон (<BODY>, <TABLE>, <TR>, <TH> або <TD>), замість параметра BGCOLOR необхідно використовувати параметр BACKGROUND, значенням якого є адреса графічного файла.

Якщо зазначене зображення менше області екрана, для якої створюється фон, то воно повторюється і по горизонталі і по вертикалі, заповнюючи повністю всю зазначену область (весь екран, таблицю або комірку у таблиці).

Наприклад, якщо файл "back.gif" містить зображення  , то у результаті виконання тега

```
<BODY BACKGROUND="back.gif">
```

весь екран має заповнитися сіткою, що складається з синіх зірочок.

Поряд з параметром BACKGROUND при завданні фону у вигляді зображення можна також використовувати параметр BGPROPERTIES зі значенням FIXED. У цьому випадку при прокручуванні сторінки фонове зображення залишиться нерухомим (ефект водяних знаків).

Можна також у тегу <BODY> задавати одночасно параметри BGCOLOR і BACKGROUND. У цьому випадку деякий час до завантаження фону-зображення фон Web-сторінки буде визначаться параметром BGCOLOR, а не фоном, заданим браузером за умовчанням, який може не гармоніювати з кольорами Web-сторінки.

Каскадні листки стилів CSS для задання фону на Web-сторінці використовують наступні властивості:

- background-color – вказує колір фону, має такі значення:
  - transparent (фон прозорий);
  - inherit – успадковує значення батька.
- background-image – задає фон у вигляді зображення, має такі значення:
  - none – фон у вигляді зображення не заданий;
  - url (ім'я графічного файла);
  - inherit – успадковує значення батька.
- background-repeat – вказує на повторюваність зображення:
  - repeat – зображення повторюється по горизонталі і по вертикалі (приймається за умовчанням);
  - repeat-x – зображення повторюється по горизонталі;
  - repeat-y – зображення повторюється по вертикалі;
  - no-repeat - зображення не повторюється;
  - inherit – успадковує значення батька.
- background-attachment – вказує властивості ковзання фону:
  - scroll – фон ковзає разом із вмістом Web-сторінки при її прокручуванні;

– fixed – фон залишається нерухомим при прокручуванні Web-сторінки (цей параметр застосовується тільки для тега <BODY>);

– inherit – успадковує значення батька.

- background-position – задає позицію фонового зображення щодо верхнього лівого краю області. Для даної властивості зазвичай задають два значення:

положення по горизонталі:

– left;

– center;

– right.

положення вертикалі:

– top;

– center;

– bottom.

Крім використання ключових слів положення також можна задавати у відсотках, пікселах або інших одиницях. Якщо застосовуються ключові слова, то порядок їх прямування не має значення, при процентному запису спочатку задається положення зображення по горизонталі, а потім, через пропуск, положення по вертикалі. Відношення між використовуваними ключовими словами і процентним записом наступне:

top left = left top = 0% 0% (у лівому верхньому куті);

top = top center = center top = 50% 0% (по центру вгору);

right top = top right = 100% 0% (у правому верхньому куті);

left = left center = center left = 0% 50% (по лівому краю і по центру);

left = left center = center left = 0% 50% (по лівому краю і по центру);

center = center center = 50% 50% (по центру);

center = center center = 50% 50% (по центру);

right = right center = center right = 100% 50% (по правому краю і по центру);

right = right center = center right = 100% 50% (по правому краю і по центру);

bottom left = left bottom = 0% 100% (у лівому нижньому куті);

bottom = bottom center = center bottom = 50% 100% (по центру внизу);

bottom = bottom center = center bottom = 50% 100% (по центру внизу);

bottom right = right bottom = 100% 100% (у правому нижньому куті).

В дужках вказано, де розташовується фоновий рисунок щодо контейнера.

- inherit – успадковує значення батька.

Для властивості background можна вказати скорочений спосіб задання значень, наприклад, background: url (bg.gif) fixed.

Використання каскадних листків стилів для задання фону на Web-сторінці показано в ***прикладі 5.1***.

5.1.3. Позиціонування елементів на Web-сторінці

Позиціонування, тобто розташування елементів на Web-сторінці здійснюється за допомогою властивості CSS position, яке має такі параметри:

– static – елементи відображаються звичайно. Використання властивостей left, top, right і bottom не призводить до якихось результатів (приймається за умовчанням);

– relative – положення елементу встановлюється щодо його початкового місця. Додавання властивостей left, top, right і bottom змінює позицію елемента і зрушує його в той чи інший бік від первинного розташування;

– absolute – вказує, що елемент абсолютно позиціонується, при цьому інші елементи відображаються на веб-сторінці немов абсолютно позіційованого елементу і немає. Положення елемента задається властивостями left, top, right і bottom, також на положення впливає значення властивості position у батьківського елементу. Так, якщо у батька значення position встановлено як static або батька немає, то відлік координат ведеться від краю вікна браузера. Якщо у батька значення position задано як fixed, relative або absolute, то відлік координат ведеться від краю батьківського елементу;

– fixed – по своїй дії це значення близько до absolute, але на відміну від нього прив'язується до зазначеної властивостями left, top, right і bottom точці на екрані і не змінює свого положення при прокрутці веб-сторінки. Браузер Firefox взагалі не відображає смуги прокручування, якщо положення елемента задано фіксованим, і воно не поміщається цілком у вікно браузера. У браузері Opera хоча і показуються смуги прокручування, але вони ніяк не впливають на позицію елемента;

– inherit – успадковує значення батька.

Розташування позіційованої елемента на екрані задається за допомогою наступних властивостей:

– left – визначає відстань від лівого краю батьківського елементу, не включаючи відступ, поле і ширину рамки, до лівого краю дочірнього елемента. Відлік координат залежить від значення властивості position. Якщо воно дорівнює absolute, як батько виступає вікно браузера і положення елемента визначається від його лівого краю. У випадку значення relative, left відраховується від лівого краю вихідного положення елемента. Якщо для батьківського елемента задано position: relative, то абсолютне позиціонування дочірніх елементів визначає їх положення від лівого краю батьків;

– right – права межа – визначає відстань від правого краю батьківського елементу, не включаючи відступ, поле і ширину рамки, до правого краю дочірнього елемента. Відлік координат залежить від значення властивості position. Якщо воно дорівнює absolute, як батько виступає вікно браузера і положення елемента визначається від його правого краю. У випадку значення relative, right відраховується від правого краю вихідного положення елемента. Якщо для батьківського елемента задано position: relative, то абсолютне позиціонування дочірніх елементів визначає їх положення від правого краю батьків;

– top – визначає відстань від верхнього краю батьківського елементу, не включаючи відступ, поле і ширину рамки, до верхнього краю дочірнього елемента. Відлік координат залежить від значення властивості position. Якщо воно дорівнює absolute, як батько виступає вікно браузера і положення елемента визначається від його верхнього краю. У випадку значення relative, top відраховується від верхнього краю вихідного положення елемента. Якщо для батьківського елемента задано position: relative, то абсолютне позиціонування дочірніх елементів визначає їх положення від верхнього краю батьків;

– bottom – визначає положення нижнього краю вмісту елемента без урахування товщини рамок і відступів. Відлік координат залежить від властивості position, воно зазвичай приймає значення relative або absolute. При відносному позиціонуванні елемента, відлік ведеться від нижнього краю вихідного положення елемента, а при абсолютному - щодо нижнього краю вікна документа. Якщо для батьківського елемента задано position: relative, то

абсолютне позиціонування дочірніх елементів визначає їх положення від нижнього краю батьків.

Якщо для одного елемента задані обидва властивості: left і right, то властивість right ігнорується. При спільному завданні властивостей top і bottom перевага віддається властивості top.

Значення властивостей left, right, top і bottom можуть задаватися:

– в одиницях довжини, допустимих в CSS;

– у відсотках від висоти батьківського елемнтов (якщо батько явно не вказаний, то в його якості виступає вікно браузера);

– auto – встановлює висоту враховуючи тип і зміст елемента (задається за замовчуванням);

– inherit – успадковує значення батька.

Якщо одиниці виміру не вказані, то мається на увазі, що вони задані в пікселях.

За допомогою CSS можна також дивись на Web-сторінці не весь елемент, а тільки його частину. Це здійснюється за допомогою властивості clip, яке працює лише для абсолютно позиціонуються елементів і може приймати такі значення:

- auto – робить видимим уесь елемент (задається за замовчуванням);
- rect (Y1, X1, Y2, X2)> – робить видимим частину зображення у вигляді прямокутника з зазначеними межами, які задаються в одиницях довжини, допустимих в CSS, і відраховуються від лівого верхнього кута зображення:
  - Y1 – верхня межа;
  - X1 – права межа;
  - Y2 – нижня межа;
  - X2 – ліва межа;
  - inherit – успадковує значення батька.

При завданні абсолютних координат для елементів Web-сторінки стає можливим накладення на екрані одного елемента на інший, тобто створення багатошарового зображення. У цьому випадку верхній браузер помістить елемент, який в HTML-документі стоїть пізніше іншого.

Разом з тим є можливість для кожного елемента задавати номер шару явно. Це робиться за допомогою властивості CSS z-index, що може приймати такі значення:

- число (позитивне, негативне і нуль). Чим більше значення, тим вище знаходиться елемент порівняно з тими елементами, у яких воно менше. При рівному значенні z-index, на передньому плані знаходиться той елемент, який в HTML-документі описаний нижче;
- auto – порядок елементів у цьому випадку задається автоматично, виходячи з їх розшатування в HTML-документі та приналежності до батьків, оскільки дочірні елементи мають той самий номер, що їх батьківський елемент (встановлюється за умовчанням);
- inherit – успадковує значення батька.

За замовчуванням усіх елементів Web-сторінці присвоюється значення z-index: 0.

Використання каскадних лстків стилів для позиціонування елемента, подання лише його частини і задання двох шарів на Web-сторінці показано в ***прикладі 5.1***.

5.2. Подання відеофрагментів і звуку на Web-сторінці

Існує декілька форматів файлів з відеофрагментами. Наприклад, розроблений фірмою MicroSoft стандарт AVI (Audio-Video Interlaced − перемежовуються аудіо-відео).

Для вказівки відеофрагмента на Web-сторінці можна використати той же тег <IMG>, що і для задання зображень, який містить такі параметри:

- DYNSRC – вказує URL-адресу файла з відеофрагментів;
- SRC – вказує URL-адресу файла із зображенням, яке виводиться на екран у тому випадку, коли відеофрагмент не може бути завантажений на Web-сторінці;
- START – вказує, як здійснюється запуск відеофрагмента:
  - fileopen – відразу після завантаження Web-сторінки;
  - mouseover – після наведення на відеофрагмент вказівника миші.
- LOOP – вказує число повторень відеофрагмента (-1 або infinite − нескінченно).

Для задання розміру та вирівнювання відеофрагментів використовуються ті ж параметри (HEIGHT, WIDTH і ALIGN), що і для зображень.

Недоліком такого підходу включення відеофрагментів на Web-сторінці є те, що його підтримує тільки браузер MicroSoft Internet Explorer. Щоб зробити висновок відеофрагментів на Web-сторінці доступним і для інших браузерів, можна скористатися тегом <EMBED>, який використовується для завантаження і відображення об'єктів (наприклад, відеофайлів, флеш-роликів, деяких звукових файлів і т.д.), які початково браузер не розуміє. Як правило, такі об'єкти потребують підключення до браузера спеціального модуля, який іменується плагін, або запуску допоміжної програми. Для тега <EMBED> обов'язковим є наявність тега, що закриває. Він містить такі параметри:

- SRC – вказує URL-адресу файла з відеофрагментів (можна вказувати файли і з іншими типами даних, але не всі з них підтримуються браузерами);
- TYPE – вказує тип MIME файла даних.
- HIDDEN – вказує, приховати об'єкт на сторінці чи ні:
  - true – приховати (зручно вказувати, наприклад, для звукових файлів);
  - false – ні (значення за замовчуванням).
- PLUGINPAGE – адреса сторінки в мережі Internet, звідки можна скачати і встановити плагін до браузера;
- ALIGN, WIDTH, HEIGHT, HSPACE і VSPACE - мають те ж призначення, що і в тегу <IMG> (див. підрозділ 5.1.1).

Тег <EMBED> не входить в специфікацію HTML 4.0 і його наявність призведе до невалідного коду.

Ця специфікація рекомендує використовувати тег <OBJECT> для завантаження зовнішніх даних замість тега <EMBED>. Однак деякі браузери не відображають, таким чином, потрібну інформацію, тому найкращим варіантом буде помістити <EMBED> всередину контейнера <OBJECT>.

Типи MIME (Multipurpose Internet Mail Extentions – багатоцільові розширення пошти) були спочатку розроблені як стандарт ідентифікації різних типів файлів для відправки їх електронною поштою, але згодом їх функції були розширені і на інші сервіси Інтернету.

У ***прикладі 5.1*** показано задання відеофрагментів на Web-сторінці як за допомогою тега <IMG>, так і за допомогою тега <EMBED>. В останньому випадку на екран виводитися не тільки відеофрагмент, а й органи керування.

Для задання звуку на Web-сторінці використовується тег <BGSOUND>, який повинен розміщуватися тільки в контейнері <HEAD> та містить такі параметри:

- SRC – вказує URL-адресу звукового файла (розширення: WAV, AU, MIDI та ін.);
- LOOP – вказує число повторень звукового супровода (-1 або infinite – нескінченно);
- BALANCE – управляє балансом звуку між правою і лівою колонками;
- VOLUME – задає гучність звучання музики.

Тег <BGSOUND> не входить в специфікацію HTML і при його використанні код не пройде валідацію. До того ж користувачі, як правило, негативно ставляться до музики, яка грає на сайті, тому використовуйте цю можливість з обережністю і за потребою.

Елемент <OBJECT> повідомляє браузеру, як завантажувати і відображати об'єкти, які початково браузер не розуміє. Як правило, такі об'єкти потребують підключення до браузера плагіну, або запуску допоміжної програми.

Специфікація HTML 4 дозволяє вкладати кілька тегів <OBJECT> з різним змістом один в одного. Це дозволяє відображати той контент, який розуміє браузер, при відсутності потрібного плагіна. Наприклад, зовнішній тег <OBJECT> завантажує відеофайл, а для випадку, коли відповідний кодек не встановлений, внутрішній тег <OBJECT> показує графічне зображення. Додатково всередину контейнера <OBJECT> можна помістити тег <PARAM>, який передає додаткові параметри для відображення об'єкта.

Тег <OBJECT> містить такі параметри:

- DATA – адреса файла для його відображення у вікні браузера (визначається щодо папки, заданої параметром CODEBASE. Якщо CODEBASE не зазначений, тоді шлях слід задавати щодо поточного документу);
- CLASSID – вказує браузеру адресу програми (програми або плагіна), яку потрібно завантажити для роботи з цим об'єктом. Можна також

використовувати ідентифікатор зареєстрованого ActiveX програми, випереджаючи його ключовим словом clsid;

- CODETYPE – вказує на тип об'єкта, який заданий параметром CLASSID;
- CODE – використовується для заміни тега <APPLET>;
- CODEBASE – шлях до папки з об'єктом, який вказаний параметром CODE або CLASSID (повинен містити тільки ім'я папки, а не файла);
- ALIGN, WIDTH, HEIGHT, HSPACE і VSPACE − мають те ж призначення, що і в тегу <IMG> (див. підрозділ 5.1.1). Тег, що закриває обов'язковий.

Тег <PARAM> призначений для передачі значень параметрів Java-аплетам або об'єктів Web-сторінки, створеним за допомогою тегів <APPLET> або <OBJECT>. Такий підхід дозволяє прямо в коді HTML-документа змінювати характеристики аплету без його додаткової компіляції. Кількість одночасно використовуваних тегів PARAM може бути більше одного і для кожного з них задається пара "ім'я/значення" через параметри NAME і VALUE:

- NAME – визначає унікальне ім'я параметра. Це ім'я має бути програмно закладено в код аплету або об'єкта;
- VALUE – значення параметра. Як значення може використовуватися рядок або число. Задання значення не є обов'язковим, оскільки для аплету або об'єкта цілком може бути достатньо одного імені, заданого параметром NAME.

Приклад 5.1

```
<HTML>
<HEAD>
<TITLE> Графіка на Web-сторінці </TITLE>
<LINK REL=STYLESHEET HREF="my.css">
<STYLE> P (color: blue; font: italic bold 8mm) </STYLE>
</HEAD>
<BODY STYLE="background:url(back.gif) fixed ">
<H1> Графіка та відеофрагменти на Web-сторінці </H1>
<TABLE WIDTH = 50% HEIGHT = 100 ALIGN = center
STYLE = "text-align: center; border: solid 3 # 00FFFF">
<TR>
```

```
      <TD STYLE="background-color:#A0FFFF"> Непрозорий фон
      <TD STYLE="background-color:transparent"> Прозорий фон
      </TABLE> <BR> <BR>
      <IMG SRC = "fish.gif" ALT = "Зображення рибок" WIDTH =
200
      ALIGN = left HSPACE = 20 STYLE = "border: groove 12
red">
      <P> Зображення збільшено в три рази, вирівняно ліворуч
від тексту, має рамку і відступи по горизонталі <BR> <BR>
<BR> <BR>
      <P ALIGN=right> Вирізаний фрагмент зображення
      <IMG SRC = "fish.gif" WIDTH = 300 STYLE = "z-index: -1;
      position: absolute; left: 500; top: 430; clip: rect
(0,250,120,100) "> <BR> <BR> <P> Відеофрагмент
      <IMG DYNSRC = "Orbit.avi" START = mouseover WIDTH = 500
      HEIGHT = 125 LOOP = 1 ALIGN = middle> вирівняний по
центру тексту, запускається з наведення миші і виконується
один раз
      <BR> <BR> <P> Відеофрагмент з органами керування
      <EMBED SRC="Orbit.avi" TYPE="video/avi">
      </BODY>
      </HTML>
```

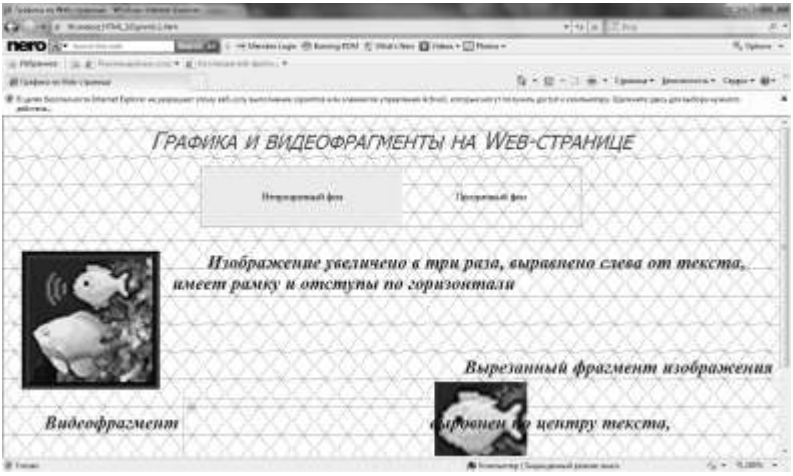


Рисунок 5.1 – Результат прикладу 5.1 на екрані

Індивідуальні завдання

Розробити HTML-документ, що задає на Web-сторінці:

- фон у вигляді зображення, який при прокручуванні сторінки залишається нерухомим;
- область з прозорим фоном і область з непрозорим фоном;
- зображення в трьох модифікаціях;
  - в рамці;
  - зі зміною розміру;
  - урізане до однієї четвертої частини з абсолютними координатами;
- відеофрагмент.

(Параметри трьох зображень і відеофрагментів для кожного варіанта наведені в таблиці 5.1).

Таблиця 5.1 – Варіанти індивідуальних завдань

№	Зображення 1		Зображення 2		Зображення 3		Відеофрагмент	
	Вирівню-вання	Рамка	Вирівню-вання	Зміна / кількість разів	Розташуван-ня рисунка	Чверть рисунка	Вирівню-вання	Запуск / кількість разів
1	2	3	4	5	6	7	8	9
1	Left	outset	right	більше/ 3	лівий/ верхній	права/ нижня	top	fileopen/2
2	Middle	inset	bottom	менше/2	правий/ верхній	ліва/ нижня	Left	mouseover/1
3	Right	ridge	top	довше 4	лівий/ нижній	права/ верхня	middle	mouseover/3
4	Bottom	groove	left	коротше /5	правий/ нижній	ліва/ верхня	right	fileopen/2
5	Top	outset	middle	більше/ 5	середина/ верх	права/ нижня	bottom	mouseover/1
6	Left	inset	right	менше/3	середина/ низ	ліва/ нижня	top	fileopen/1
7	middle	ridge	bottom	довше/6	лівий/ верхній	права/ верхня	left	mouseover/3

Продовження табл. 5.1

1	2	3	4	5	6	7	8	9
8	Right	groove	top	коротше/2	правий/ верхній	ліва/ верхня	middle	fileopen/2
9	Bottom	outset	left	більше/5	лівий/ нижній	права/ нижня	right	mouseover/1
10	Top	inset	middle	менше/4	правий/ нижній	ліва/ нижня	bottom	fileopen/3
11	Left	ridge	right	довше/8	середина/ верх	права/ верхня	top	mouseover/1
12	middle	groove	bottom	коротше/4	середина/ низ	ліва/ верхня	left	fileopen/4
13	right	outset	top	більше/6	лівий/ верхній	права/ нижня	middle	mouseover/5
14	bottom	inset	left	менше/2	правий/ верхній	ліва/ нижня	right	fileopen/2
15	Top	ridge	middle	довше/7	лівий/ нижній	права/ верхня	bottom	mouseover/–1
16	Left	groove	right	коротше/5	правий/ нижній	ліва/ верхня	top	fileopen/2
17	middle	outset	bottom	більше/3	середина/ верх	права/ нижня	left	mouseover/–1
18	right	inset	top	менше/4	середина/ низ	ліва/ нижня	middle	fileopen/3
19	bottom	ridge	left	довше/8	лівий/ верхній	права/ верхня	right	mouseover/4
20	Top	groove	middle	коротше/6	правий/ верхній	ліва/ верхня	bottom	fileopen/1

ЛАБОРАТОРНА РОБОТА 6
РОБОТА З ПОСИЛАННЯМИ НА WEB-СТОРІНЦІ

Мета: Вивчення засобів і можливостей HTML і CSS для подання на Web-сторінці посилань і сегментованої графіки.

6.1. Подання посилань на Web-сторінці

Посилання – це виділені об'єкти на Web-сторінці (слова, зображення та ін.), які при наведенні курсору миші змінюють свої параметри з появою на екрані зображення "долоньки". Після клацання миші по посиланню відбувається пошук і завантаження необхідної Web-сторінки.

6.1.1. Задання параметрів посилань

Для задання посилання і його параметрів використовується тег-контейнер <A> посилання </A> (Anchor – якір), який містить такі параметри:

- HREF – вказує URL-адресу Web-сторінки, яка буде завантажена після клацання миші за посиланням;
- NAME – вказує ім'я в документі, куди буде переміщено вказівник при внутрішньому використанні посилань (див. пункт 1.4);
- TARGET – вказує ім'я вікна (фрейму), куди буде завантажена викликана Web-сторінка (див. розділ 7.1.2 опису лабораторної роботи 7);
- TITLE – задає пояснювальний напис під посиланням, який з'являється через якийсь час після наведення миші на посилання.

Обов'язковими є тільки параметри HREF і NAME, які разом в одному тезі використовуватися не можуть.

Адреса посилання може бути абсолютною і відносною. Абсолютні адреси працюють скрізь і всюди незалежно від імені сайту або Web-сторінки, де прописане посилання. Відносні посилання, як випливає з їх назви, побудовані щодо поточного документа або кореня сайту.

Наприклад, для задання посилання зі стандартними параметрами, за допомогою якого виконується завантаження Web-сторінки з описом лабораторної роботи 1, використовується такий тег:

```
<A HREF="lab1.htm"> Лабораторна робота 1 </A>
```

Для зміни параметрів кольору посилання використовують такі параметри тега <BODY>:

- LINK – задає колір невідвіданого посилання (за умовчанням синій);
- VLINK – задає колір відвіданого посилання (за умовчанням чорний);
- ALINK – задає колір посилання після клацання миші по ньому, але до завантаження Web-сторінки (використовується рідко).

Посилання може бути задано також за допомогою рисунка. Для цього необхідно тег <IMG>, який задає рисунок, помістити всередині контейнера <A> </A>:

```
<A HREF="lab2.htm"> <IMG SRC="fish.gif"> </A>
```

У **прикладі 6.1** наведено використання стандартних посилань, а також посилань у вигляді зображення.

6.1.2. Використання CSS у посиланнях

У каскадних листках стилів для посилань є псевдокласи з таким форматом:

- A:link{} – для задання параметрів (кольору, форми та ін.) тих посилань, які ще не були використані для завантаження Web-сторінок;
- A:visited{} – для задання параметрів тих посилань, які вже були використані для завантаження Web-сторінок;
- A:active{} – для задання параметрів тих посилань, які були використані, але завантаження Web-сторінки ще не відбулось;
- A:hover{} – для задання параметрів тих посилань, на які наведено курсор миші.

Псевдокласи та їх властивості описуються там же, де і класи (див. зовнішнє і внутрішнє задання CSS у лабораторній роботі 2).

Якщо на Web-сторінці потрібно використовувати декілька зображень для посилань, необхідно разом з псевдокласами задавати звичайні класи. Наприклад, можна створити клас зображень посилань у вигляді кнопок. Нижче наводяться каскадні листки стилів, які описують клас посилань "kn", що задає для трьох станів посилань (link, visited і hover) такі параметри, як колір, тип і обсяг тексту, тип стану кнопки (відтиснута – outset, натиснута – inset) і колір фону:

```
A.kn: link {color: #004040; font: italic 0.6cm; border:
outset #FF00AA 5; background: #E0E0E0; text-decoration: none}
A.kn: visited {color: #0000A0; font: italic 0.6cm;
border: outset #FF00AA 5; background: #00E0E0; text-
decoration: none}
A.kn: hover {color: white; font: italic 0.6cm; border:
inset #FF00AA 5; background: #FF60a0; text-decoration: none}
```

Ці описи параметрів посилань додані в зовнішній файл CSS "my.css" (тепер він називається "my_d.css") і використовуються у ***прикладі 6.1*** для задання параметрів посилань за допомогою каскадних листків стилів.

Посилання класу "kn" задається так:

```
<A CLASS = "kn". . . >
```

6.1.3. Зображення курсору на Web-сторінці

При наведенні курсору миші на посилання зазвичай з'являється зображення "долоньки". У CSS є можливість змінити вигляд курсору (і не тільки для посилань, але і для інших елементів Web-сторінки, наприклад, зображень). Для цього необхідно використовувати властивість cursor, яка встановлює форму курсора, коли він знаходиться в межах елемента. Вид курсору залежить від операційної системи і встановлених параметрів:

- url (URL-адрес) – дозволяє встановити свій власний курсор, для цього потрібно вказати шлях до файлу, в якому вказана форма курсора;
- auto – вид курсора за замовчуванням для поточного елемента;
- inherit – успадковує значення батька;
- default – зображення, якє задається за умовчанням (звичайно це "долонька");
- hand – "долонька";
- text – текстовий курсор;
- help – у вигляді знака питання ("?");
- wait – у вигляді пісочного годинника (означає "почекайте");
- e-resize або w-resize – подвійна горизонтальна стрілка (схід – захід);
- ne-resize або sw-resize – подвійна стрілка напрямку північний схід – південний захід;

- nw-resize або se-resize – подвійна стрілка напрямку північний захід – південний схід;
- s-resize або n-resize – вертикальна подвійна стрілка (північ – південь).

У ***прикладі 6.1*** показано використання зображення курсору у вигляді знака питання.

6.1.4. Сегментована графіка

До сегментованої графіки належать зображення, які умовно розбиті на окремі частини у вигляді певних геометричних фігур. Кожна така фігура являє собою область чутливості до клацання миші і є посиланням.

Для перерахування областей чутливості використовується тег-контейнер <MAP>, параметр якого NAME задає ім'я карти з описом цих областей.

Кожна область чутливості описується за допомогою одиночного тега <AREA>, який містить такі параметри:

- SHAPE – ставить геометричну фігуру, яка описує форму області чутливості:
  - rect – прямокутник;
  - circ – коло;
  - poly – багатокутник.
- COORDS - ставить координати цих фігур:
  - для прямокутника – у вигляді координат його лівого правого кута і лівого нижнього (x1, y1, x2, y2);
  - для окружності – у вигляді пари координат центра і величини радіуса (x, y, r);
  - для багатокутника – у вигляді пар координат для кожної вершини (x1, y1, x2, y2,..).
- HREF – задає URL-адресу потрібної Web-сторінки;
- NOHREF – указує на те, що дана область нечутлива до наведення мишки (повинна бути оголошена першої);
- ALT – задає пояснювальний напис під посиланням, який з'являється через якийсь час після наведення курсору мишки на посилання (аналогічно параметру TITLE тега <A>).

Наприклад, для задання фігур і координат трьох областей чутливості (вони виділені кольором) зображення (вставити рисунок) використовуються такі значення параметрів SHAPE і COORDS:

- для прямокутника - SHAPE = rect COORDS = 10,6,50,30
- для трикутника - SHAPE = poly COORDS = 10,40,10,80,50,40
- для окружності - SHAPE = CIRC COORDS = 88,49,20

Щоб зв'язати між собою рисунок із сегментованою графікою, заданий тегом <IMG> з картою зображення, заданий контейнером <MAP>. . . </ MAP>, необхідно вказати в тезі <IMG> параметр usemap, значенням якого є значення параметра name тега <MAP> з доданим першим символом решітки (#).

У ***прикладі 6.1*** показано використання сегментованої графіки на Web-сторінці.

6.1.5. Використання локальних посилань

Локальні посилання використовуються для полегшення доступу до окремих частин документа, поданого на Web-сторінці. Для цього в HTML-документі за допомогою тега <A> в необхідних місцях документа встановлюються мітки доступу. При цьому у тегу <A> замість параметра HREF використовується параметр NAME, який задає ім'я мітки доступу. До цього місця на Web-сторінці можна звернутися за допомогою звичайного посилання, на якій як значення параметра HREF вказується ім'я мітки з знаком #, що передує.

Локальні посилання можна також використовувати для доступу до внутрішніх частин іншого документа. У цьому випадку до URL-адреси документа додається знак # і ім'я внутрішнього посилання у цьому документі. Наприклад, за допомогою посилання

```
<A HREF="lab2.htm#2">
```

забезпечується доступ не до початку опису лабораторної роботи 2, а до її розділу 2.

У ***прикладі 6.1*** показано застосування локальних посилань для внутрішнього і зовнішніх документів.

Приклад 6.1

```
<HTML>
<HEAD>
<TITLE> Ссылки на Web-странице </TITLE>
<LINK REL=stylesheet HREF="my_d.css">
<STYLE> P, TABLE (color: # 0000c0; font: 6mm) </STYLE>
</HEAD>
<BODY>
<A NAME="beg"> </A> <BR>
<TABLE WITH=80% HEIGHT=130% ALIGN=center>
<TR> <TD> Ссылки со стандартными параметрами:
<TD>
<A HREF="lab1.htm"> Лабораторная работа № 1 </A>
<TR> <TD> Локальная ссылка:
<TD> <A HREF="#text"> Текст HTML-документа </A>
<TR> <TD> Ссылка в виде рисунка:
<TD> <A HREF="prim5-1.htm"
TITLE="Использование графики в ссылках">
<IMG SRC="fish.gif"> </A>
<TR> <TD> Ссылка в виде кнопки на раздел 2 документа:
<TD> <A CLASS="kn" HREF="lab2.htm#2">
Лабораторная работа № 2 </A>
<TR> <TD> Ссылка-справка:
<TD>
<A CLASS="kn"
STYLE = "border-style:none; cursor:help" HREF="prim3-
1.htm"> Списки </A>
<TR> <TD> ссылка в виде сегментированной графики:
<TD> <IMG SRC="seg.bmp" ALIGN=middle BORDER=0
USEMAP="#my_map">
<TR> <TD> Ссылка в виде кнопки-изображения:
<TD> <A HREF="prim4-1.htm" CLASS="kn" style="border-
width:10">
<IMG DYNSRC="Orbit.avi" START=fileopen LOOP=1 HEIGHT=40>
</A>
</TABLE>
```

```
<MAP NAME="my_map">
<AREA SHAPE=circ COORDS=88,49,5 NOHREF>
<AREA SHAPE=rect COORDS=10,6,50,30 HREF="prim1-2.htm"
ALT="Задание параметров шрифтов">
<AREA SHAPE=poly COORDS=10,40,10,80,50,40 HREF="prim1-
3.htm" ALT="Физическое и логическое форматирование текста">
<AREA SHAPE=CIRC COORDS=88,49,20 HREF="prim2-1.htm"
ALT="Использование CSS для форматирования текста">
</MAP>
<A NAME="text"> </A> <BR>
<A CLASS="kn" HREF="#beg"> Начало </A>
<P> Текст HTML-документа
</BODY>
</HTML>
```

Рисунок 6.1 – Результат прикладу 6.1 на екрані

Індивідуальні завдання

Розробити HTML-документ, який задає на Web-сторінці:

- звичайні посилання;
- два локальних посилання;
- посилання у вигляді сегментованої графіки з двумя областями чутливості;
- курсор з необхідним зображенням

(параметри всіх трьох посилань і курсор наведені у таблиці 6.1).

Таблиця 6.1 – Варіанти індивідуальних завдань

№	Посилання у вигляді	Локальні посилання у вигляді	Локальні посилання у вигляді	Області чутливості сегментованої графіки	Вид курсору
1	2	3	4	5	6
1	зображення	кнопки	тексту	3-кутник, квадрат	text
2	кнопки	тексту	зображення	квадрат, коло	help
3	кнопки з зображ.	кнопки	тексту	коло, 5-кутник	wait
4	тексту	кнопки	зображення	4-кутник, коло	e-resize
5	зображення	тексту	кнопки	5-кутник, квадрат	n-resize
6	кнопки	зображення	тексту	3-кутник, 6-кутник	ne-resize
7	кнопки з зображ.	кнопки	тексту	квадрат, 5-кутник	nw-resize
8	тексту	кнопки з зображ.	зображення	коло, квадрат	text
9	зображення	кнопки з зображ.	тексту	4-кутник, 5-кутник	help
10	кнопки	кнопки з зображ.	тексту	5-кутник, 4-кутник	wait
11	кнопки з зображ.	тексту	зображення	3-кутник, 5-кутник	e-resize
12	тексту	зображення	кнопки	квадрат, 6-кутник	n-resize

Продовження таблиці 6.1

1	2	3	4	5	6
13	зображення	кнопки	тексту	коло, квадрат	ne-resize
14	кнопки	тексту	зображення	4-кутник, 4-кутник	nw-resize
15	кнопки з зображ.	тексту	зображення	5-кутник, коло	text
16	тексту	кнопки	кнопки з зображ.	3-кутник, коло	help
17	зображення	тексту	кнопки з зображ.	квадрат, коло	wait
18	кнопки	тексту	кнопки з зображ.	коло, 4-кутник	url
19	кнопки з зображ.	кнопки	тексту	5-кутник, квадрат	n-resize
20	тексту	кнопки з зображ.	кнопки	5-кутник, 4-кутник	ne-resize

ЛАБОРАТОРНА РОБОТА 7
ВИКОРИСТАННЯ ФРЕЙМІВ НА WEB-СТОРІНЦІ

Мета: вивчення засобів і можливостей HTML для організації за допомогою фреймів багатовіконному роботи на Web-сторінці.

7.1. Подання фреймів на Web-сторінці

7.1.1. Задання фреймової структури

Щоб зробити перегляд документів на Web-сторінці більш зручним, можна скористатися багатовіконним інтерфейсом, реалізованим за допомогою фреймів. У цьому випадку можна завантажити відразу кілька документів (Web-сторінок) і працювати одночасно з усіма ними.

На відміну від звичайного HTML-документа у документі з описом фреймів немає тега-контейнера <BODY>…</BODY>. Замість нього використовується тег-контейнер <FRAMESET>…</FRAMESET>, який ділить екран на декілька горизонтальних частин (вікон), або на декілька вертикальних вікон. Кожне з вікон описується у вигляді фрейму з допомогою тега <FRAME>. Тег <FRAMESET> містить такі параметри:

- COLS – вказує через кому ширину вертикальних вікон у пікселях або у % від ширини екрана (якщо задається *, то цьому вікну відводиться частина, яка залишилася від розміру екрана);
- ROWS – вказує через кому висоту горизонтальних вікон у пікселях або у % від висоти екрана (якщо задається *, то цьому вікну відводиться частина, яка залишилася від розміру екрана);
- FRAMEBORDER:
  - 1 – фрейми мають рамку;
  - 0 – фрейми не мають рамку.
- FRAMESPACING – вказує відстань між фреймами в пікселях.

Наприклад, тег

```
<FRAMESET COLS="20%,30%,*">
<FRAME…>
<FRAME…>
<FRAME…>
</FRAMESET>
```

ділить екран на три вертикальних вікна, які займають відповідно 20 %, 30 % і частину, яка залишилася (50 %) екрана. На місці кожного з тегів <FRAME> може бути вказаний тег <FRAMESET>…</ FRAMESET>. Це дозволяє формувати на екрані складну багатовіконну структуру.

У ***прикладі 7.1*** показано побудову та використання тривіконної структури на Web-сторінці.

7.1.2. Задання параметрів фреймів

Для опису характеристик кожного вікна використовується тег <FRAME>, який містить наступні параметри:

- MARGINHEIGHT – вказує відступ у пікселях по висоті від меж фрейму;
- MARGINWIDTH – вказує відступ у пікселях по ширині від меж фрейму;
- FRAMEBORDER – має ті ж значення (0,1), що і однойменний параметр тега FRAMESET;
- NAME – вказує ім'я фрейму, яке може бути використане у посиланнях для завантаження Web-сторінки на потрібне вікно екрана (див. опис параметра TARGET у розділі 6.1.1 опису лабораторної роботи 6); є ряд стандартних імен вікон:
  - _self – для завантаження Web-сторінки в поточне вікно (приймається за замовчуванням, якщо на засланні не заданий параметр TARGET);
  - _top – для завантаження Web-сторінки на весь екран;
  - _blank – для завантаження Web-сторінки в нове вікно (відрізняється від _top тим, що не можна повернути у вихідне вікно по стрілці "Назад").
- NORESIZE – забороняє змінювати за допомогою мишки розміри фрейму;
- SCROLING – вказує на один з трьох режимів перегляду фрейму:
  - yes – зі створенням смуги прокручування;
  - no – без створення смуги прокручування;
  - auto – зі створенням смуги прокручування тільки тоді, коли файли даних не поміщаються у вікні фрейму (приймається за замовчуванням).

- SRC – URL-адреса Web-сторінки, що завантажується у вікно фрейму;
- BORDERCOLOR – вказує колір межі фрейму (ширина межі задається параметром FRAMESPACING).

7.2. Плаваючі фрейми

При створенні багатовіконної Web-сторінки крім застосування фреймової структури на базі тега <FRAMESET>…</FRAMESET> (див. пункт 7.1.1) також можуть бути використані плаваючі фрейми. У цьому випадку в HTML-документі замість тега <FRAMESET>…</FRAMESET> використовується, як і у всіх інших Web-сторінках, тег <BODY>…</BODY>.

Плаваючий фрейм, подібно зображенню, буде розташовуватися на екрані у тому місці, яке відповідає розташуванню тега з його описом у HTML-документі. Для задання плаваючого фрейму використовується тег-контейнер <IFRAME>…</IFRAME>. Оскільки плаваючий фрейм багато в чому має властивості і фрейму, і зображення, то він має частину параметрів (MARGINHEIGHT, MARGINWIDTH, NAME, SCROLING, SRC, BORDERCOLOR), які властиві фреймам і розглянут у підрозділі 7.1.2, інша частина його параметрів (ALIGN, HEIGHT, WIDTH, HSPACE, VSPACE) характерна для зображень і розглянута в лабораторній роботі 5 у підрозділі 5.1.1.

Нижче наведено приклад задання плаваючого фрейму, з ім'ям "ff", розміром 200*200 пікселів, до якого завантажується Web-сторінка з адресою "prim3-1.htm":

```
<IFRAME NAME="ff" SRC="prim3-1.htm" HEIGHT=200 WIDTH=200> </IFRAME>
```

У прикладі 1 поряд зі звичайними фреймами показане застосування плаваючих фреймів (у файлі "left.htm").

7.3. Розробка багатовіконної Web-сторінки

У ***прикладі 7.1*** показано побудову тривіконної Web-сторінки, яка має відповідно три фрейми: один горизонтальний ("title") і два вертикальних ("left" і "right").

Приклад 7.1

```
<HTML>
<HEAD>
<TITLE> Фреймы на Web-странице </TITLE>
</HEAD>
<FRAMESET ROWS="60,*" FRAMESPACING=0>
<FRAME NAME="title" SRC="title.htm" NORESIZE
SCROLLING="no"
MARGINHEIGHT=0 MARGINTEIGHT=0
STYLE="border:red 6 dashed">
<FRAMESET COLS="30%,*" FRAMESPACING=2>
<FRAME NAME="left" SRC="left.htm" FRAMEBORDER=0>
<FRAME NAME="right" SRC="lab1.htm" FRAMEBORDER=1
BORDERCOLOR=#0000E0>
</FRAMESET>
</FRAMESET>
</HTML>
```

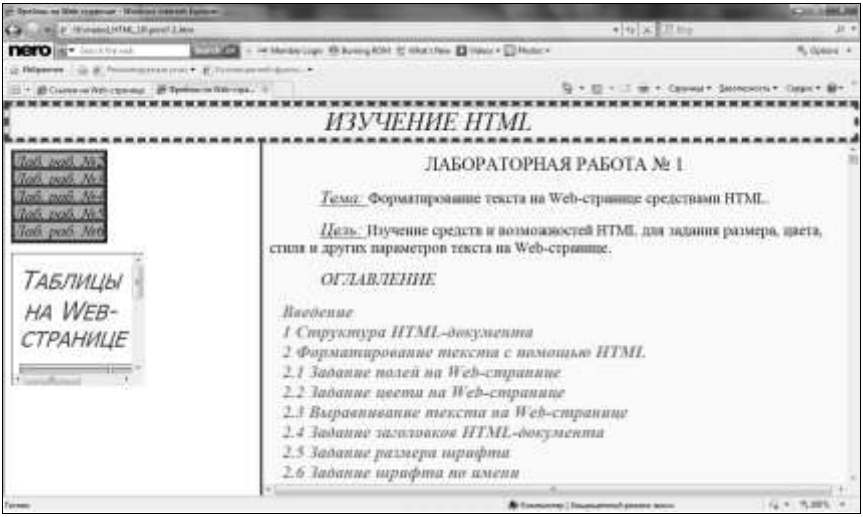


Рисунок 7.1 – Результат прикладу 7.1 на екрані

Як видно з фреймової структури, екран спочатку ділиться на два горизонтальних вікна: перше вікно має розмір 60 пікселів, друге займає

частину, яка залишилася від розміру екрана. Потім друге горизонтальне вікно ділиться на два вертикальних з розмірами відповідно 30 % і 70 % екрану.

7.3.1. Фрейм "title"

Фрейм "title" використовується для задання загальних заголовків або заголовків відомостей. Може бути використаний як домашня сторінка (home page). У ньому заборонені смуги прокручування і зміна розміру вікна.

Файл "title.htm", який завантажується у вікно "title", містить HTML-документ з двох рядків, які задають колір фону, параметри напису і сам напис:

```
<P STYLE="background:#E0FFE0; color:#0000C0;font: italic
10mm; text-align:center"> ВИВЧЕННЯ HTML
```

7.3.2. Фрейм "left"

Файл "left.htm", який завантажується у вікно "left", містить HTML-документ, у якому заданий набір посилань у вигляді кнопок, що здійснюють доступ до описів лабораторних робіт, та плаваючий фрейм:

```
<HTML>
<HEAD>
<LINK REL=stylesheet HREF="my_d.css">
</HEAD>
<BODY>
<A CLASS="kn" HREF="lab2n.htm" TARGET="right"> Лаб.роб. № 2
</A> <BR>
<A CLASS="kn" HREF="lab3n.htm" TARGET="right"> Лаб.роб. № 3
</A> <BR>
<A CLASS="kn" HREF="lab4n.htm" TARGET="right"> Лаб.роб. № 4
</A> <BR>
<A CLASS="kn" HREF="lab5n.htm" TARGET="_top"> Лаб.роб. № 5
</A> <BR>
<A CLASS="kn" HREF="lab6n.htm" TARGET="ff"> Лаб.роб. № 6
</A> <BR> <BR>
<IFRAME NAME="ff" SRC="prim2-2.htm" HEIGHT=200 WIDTH=200>
</IFRAME>
</BODY>
</HTML>
```

Перші три посилання завантажують Web-сторінки з описами лабораторних робіт 2, 3 та 4 під рамку з ім'ям "right", четверте посилання завантажує Web-сторінку з описом лабораторної роботи 5 на весь екран, а останне посилання завантажує Web-сторінку з описом лабораторної роботи 6 у плаваючий фрейм з ім'ям "ff". Опис параметрів плаваючого фрейму і його використання на Web-сторінці наведено у розділі 7.2.

Індивідуальні завдання

Розробити HTML-документ, який задає на Web-сторінці:

- чотиривіконну фреймову структуру, топологія якої і номери фреймів наведені у таблиці 7.1;
- у фреймі № 1 зазначається прізвище розробника;
- у фреймі № 2 поміщаються три посилання:

– по першим (оформлене у вигляді тексту) здійснюється завантаження опису лабораторної роботи 2 на фрейм № 3

– за другим (оформлене у вигляді тексту) здійснюється завантаження опису лабораторної роботи 3 на фрейм № 4

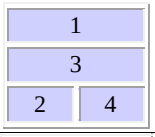
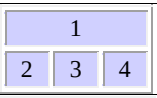
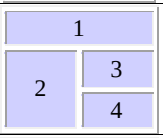
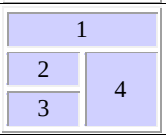
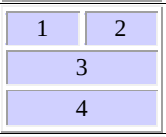
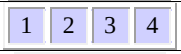
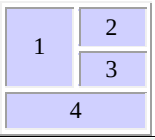
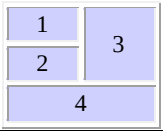
– за третит (оформлене у вигляді кнопки) здійснюється завантаження опису поточної лабораторної роботи в стандартне вікно, тип якого зазначений у таблиці 7.1.;

- плаваючий фрейм, в який завантажується Web-сторінка з прикладом prim3-1.htm.

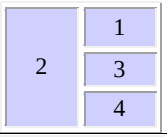
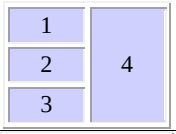
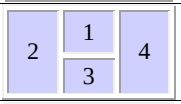
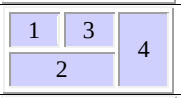
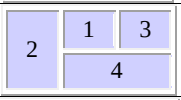
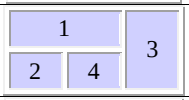
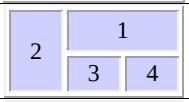
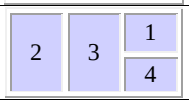
Таблиця 7.1 – Варіанти індивідуальних завдань

№	Тип вікна	Фреймова структура
1	2	3
1	_blank	

Продовження таблиці 7.1

1	2	3
2	_self	
4	_blank	
5	_self	
6	_top	
7	_blank	
8	_self	
9	_top	
10	_blank	
11	_self	
12	_top	

Закінчення табл. 7.1

1	2	3
13	_blank	
14	_self	
15	_top	
16	_blank	
17	_self	
18	_top	
19	_blank	
20	_self	

ЛАБОРАТОРНА РОБОТА 8
ЗАДАННЯ ФОРМ НА WEB-СТОРІНЦІ

Мета: вивчення засобів і можливостей HTML для створення форм на Web-сторінці.

8.1. Використання технології "клієнт-сервер" при розробці Web-сторінок

Технологія "клієнт-сервер" передбачає поділ прикладної програми на дві частини – клієнтську, яка виконується на боці клієнта, і серверну, яка виконується на боці сервера.

До цих пір при вивченні HTML розглядалися питання розробки Web-сторінок та їх інтерпретації за допомогою браузера на боці клієнта. Тому з точки зору технології "клієнт-сервер" створені Web-сторінки є клієнтськими додатками. Всі зміни на цих сторінках відбуваються тільки при завантаженні нових Web-сторінок, тому вони називаються статичними (пасивними).

Щоб зробити Web-сторінку активною (динамічною), необхідно розробити програму, яка виконується на боці сервера, за допомогою якої відбувається обробка і зміна змісту Web-сторінки.

Одним із засобів, які дозволяють здійснити взаємодію між клієнтською і серверною частинами програми, є інтерфейс CGI (Common Gateway Interface – стандартний шлюзовий інтерфейс), що має реалізації як для Windows-орієнтованих програм, так і для додатків, що функціонують в середовищі Unix. Цей інтерфейс розроблений так, щоб для створення серверного додатка можна було використовувати будь-яку мову програмування, який може працювати зі стандартними пристроями введення /виводу.

Серверна частина програми, що обробляє дані клієнта, які получені з використаням CGI-інтерфейсу, називається CGI-додатком.

При використанні CGI-додатків необхідна інформація на Web-сторінці клієнта, яка призначена для передачі на сервер та подається у вигляді форм.

8.2. Створення форми на Web-сторінці

8.2.1. Параметри тега <FORM>

Для створення форми на Web-сторінці в HTML-документі використовується тег-контейнер <FORM>…</FORM>. Форма призначена для обміну даними між користувачем і сервером. Область застосування форм не обмежена відправкою даних на сервер, за допомогою клієнтських скриптів можна отримати доступ до будь-якого елементу форми, змінювати його і застосовувати на власний розсуд.

Документ може містити будь-яку кількість форм, але одночасно на сервер може бути відправлена тільки одна форма. З цієї причини дані форм повинні бути незалежні один від одного.

Для відправки форми на сервер використовується кнопка Submit, того ж можна добитися, якщо натиснути клавішу Enter в межах форми. Якщо кнопка Submit відсутня в формі, клавіша Enter імітує її використання.

Тег <FORM> може мати теги <INPUT>, <TEXTAREA> і <SELECT>, які призначені для створення різних органів керування форми (полів уведення даних, кнопок, перемикачів).

Тег <FORM> містить такі параметри:

- NAME − визначає унікальне ім'я форми. Як правило, ім'я форми використовується для доступу до її елементів через скрипти;
- ACTION – вказує URL-адресу програми (CGI-додатки), яка виконує обробку даних форм. В якості обробника може виступати CGI-програма або HTML-документ, який включає в себе серверні сценарії. Після виконання обробником дій по роботі з даними форми він повертає новий HTML-документ. Якщо параметр ACTION відсутній, поточна сторінка перезавантажується, повертаючи всі елементи форми до їх значень за замовчуванням;
- METHOD – вказує один з двох основних способів передачі даних від форми на сервер:
  - GET – CGI-додаток отримує дані з форми на Web-сторінці через змінну середовища з ім'ям QUERY_STRING (за замочуванням);
  - POST – CGI-додаток отримує дані з форми на Web-сторінці через стандартний потік вводу;
- ENCTYPE – вказує MIME-тип переданих даних;

- TARGET – визначає вікно (наприклад, "windowName"), в яке завантажується результат передачі і обробки форми. Коли використовується форма з атрибутом TARGET, сервер помещаает відповіді у вікно windowName замість вікна, що містить форму. windowName може бути існуючим вікном, ім'ям фрейма, визначеного в тезі <FRAMESET> або одним з імен фрейму: _top, _parent, _self або _blank, але не може бути виразом JavaScript.

Метод GET є одним з найпоширеніших і призначений для отримання необхідної інформації і передачі даних в адресному рядку. Пари "ім'я = значення" приєднуються в цьому випадку до адреси після знаку питання і розділяються між собою амперсандом (символ &). Зручність використання методу GET полягає в тому, що адресу з усіма параметрами можна використовувати неодноразово, зберігши його, наприклад, в "Вибране" браузера, а також змінювати значення параметрів прямо в адресному рядку.

Метод POST посилає на сервер дані в запиті браузера. Це дозволяє відправляти більшу кількість даних, аніж доступно методу GET, оскільки у нього встановлено обмеження в 4 КБ. Великі обсяги даних використовуються в форумах, поштових службах, заповненні бази даних і т. д.

8.2.2. Параметри тега <INPUT>

Тег <INPUT> дозволяє створювати різні елементи інтерфейсу і забезпечити взаємодію з користувачем. Головним чином, <INPUT> призначений для створення текстових полів, різних кнопок, перемикачів і прапорців. Хоча елемент <INPUT> не потрібно в загальному випадку поміщати всередину контейнера <FORM>, що визначає форму, проте якщо введені користувачем дані повинні бути відправлені на сервер, де їх обробляє CGI-програма, то вказувати FORM обов'язково. Так само йде справа і в разі обробки даних за допомогою клієнтських додатків, наприклад, скриптів на мові JavaScript.

Тег <INPUT> містить такі параметри:

- TYPE – вказує тип органу керування, його значеннями є:
  - text – (значення за замовченням) текстове поле для введення символів з клавіатури, обсяг поля та максимальне число символів, що вводяться, визначаються значеннями параметрів SIZE і MAXLENGTH тега <INPUT>;

– password – поле для введення пароля аналогічне параметру text, але текст, уведений користувачем відображається на екрані у вигляді зірочок, щоб приховати пароль при введені;

– checkbox – незалежний перемикач (прапорець), дозволяє вибрати один або більш одного варіанту із запропонованих;

– radio – перемикач, що залежить від стану інших перемикачів цього типу, використовується для вибору тільки одного значення з декількох; тому всі перемикачі, що задають можливі значення для вибору, повинні мати однакове ім'я (за замовчуванням має значення "on");

– file – поле для введення імені файлу, який пересилається на сервер;

– button – кнопка, напис задається параметром VALUE;

– reset – кнопка для встановлення значень усіх органів керування форми в початковий стан, напис задається параметром VALUE (за замовчуванням "Скинути");

– submit – кнопка для відправлення із заповненої форми Web-сервера, напис задається параметром VALUE (за замовчуванням "Подача запиту");

– image – має таке ж значення, що й параметр submit, але реалізований у вигляді зображення, в цьому випадку на сервер передаються також координати (з іменами x і y) натискання за рисунком відносно його лівого верхнього кута;

– hidden – приховане поле, не відображається на екрані, дозволяє передавати на сервер нередаговані дані (наприклад, розмір переданого файлу).

- NAME - вказує ім'я органу керування, надсилається на сервер програмі обробки форми разом з його значеням;
- VALUE – вказує значення або стан органу керування, заданий початково;
- CHECKED – використовується для встановлення початкового стану перемикача; (если параметр не указан - берется первое значение);
- SIZE – вказує (числом символів) ширину поля введення текстової інформації, за умовчанням приймається рівним 20;
- MAXLENGTH – вказує максимальну кількість символів, яка може бути введена у даному полі, за умовчанням не має обмежень;

- ALIGN – вказує вирівнювання тексту біля форми;
- SRC – URL-адреса графічного файла для кнопки у вигляді зображення, (тип image);
- ALIGN – вказує на вирівнювання зображення;
- ALT – альтернативний текст для кнопки із зображенням;
- BORDER – вказує товщину рамки навколо зображення;
- DISABLED – блокує доступ до елементу і його зміну (за замовчуванням значення вимкнено);
- READONLY – встановлює, що поле не може змінюватися користувачем (за замовчуванням значення вимкнено). Проте, стан і зміст поля можна міняти за допомогою скриптів.

У ***прикладі 8.1*** показано задання полів форми на Web-сторінці за допомогою тега <INPUT>.

8.2.3. Параметри тега <TEXTAREA>

Тег <TEXTAREA> є елементом форми для створення області, в яку можна вводити кілька рядків тексту. На відміну від тега <INPUT> в текстовому полі допустимо робити переноси рядків, вони зберігаються при відправленні даних на сервер.

Між тегами <TEXTAREA> і </ TEXTAREA> можна помістити будь-який текст, який буде відображатися усередині поля.

Має наступні параметри:

- NAME – вказує ім'я багаторядкового поля, надсилається програмі обробки форми;
- ROWS – вказує розмір поля по вертикалі (в рядках);
- COLS – вказує розмір поля по горизонталі (у символах);
- WRAP – вказує тип переносу слів, має такі значення:
  - off – означає, що текст в полі не згортається, тобто не переноситься на новий рядок, а триває за межею поля;
  - soft – текст в поле згортається, але на сервер буде передаватися без згортання;
  - hard – текст в поле згортається і на сервер буде передаватися з включеним символом нового рядка (виконується за замовчуванням);
- DISABLED – блокує доступ до елементу і його зміну (за замовчуванням значення вимкнено);

У ***прикладі 8.1*** показано використання многорядкового текстового поля на Web-сторінці за допомогою тега <TEXTAREA>.

8.2.4. Параметри тега <SELECT>

Тег <SELECT> дозволяє створити елемент інтерфейсу у вигляді списку, а також список з одним або множинним вибором. Кінцевий вигляд залежить від використання параметра SIZE тега <SELECT>, який встановлює висоту списку. Ширина списку визначається самим широким текстом, зазначеним в тегах <OPTION> (може змінюватися за допомогою стилів).

Кожен пункт створюється за допомогою тега <OPTION>, який повинен бути вкладений в контейнер <SELECT>. . . </ SELECT>. Якщо планується відправляти дані списку на сервер, то потрібно помістити елемент <SELECT> всередину форми. Це також необхідно, коли до даних списку йде звернення через скрипти.

Для тега <SELECT> визначені такі параметри:

- NAME – вказує ім'я списку, яке разом з обраним рядком або рядками передається серверу або використовується в скриптах;
- SIZE – вказує кількість рядків списку, які відображаються на екрані, інші згорнуті і для появи на екрані вимагають прокручування списку.
- DISABLED – блокує доступ до елементу і його зміну (за замовчуванням значення вимкнено);
- MULTIPLE – вказує на те, що можуть бути відзначені і передані на сервер відразу кілька рядків (їх відмітка здійснюється утриманням клавіші CTRL).

Кожен рядок списку всередині контейнера <SELECT>…</SELECT> задається за допомогою тега <OPTION>, який містить такі параметри:

- SELECTED – зазначає спочатку вибрані пункти списку;
- VALUE – задає значення вибраного пункта списка.

Нижче наводиться приклад використання тега <SELECT>, у якому задається набір з трьох рядків.

```
<SELECT NAME="number" SIZE=2>
<OPTION VALUE="first"> Example_1
<OPTION SELECTED VALUE="second"> Example1_2
<OPTION> Example1_3
</SELECT>
```

Оскільки параметр SIZE дорівнює двом, на екрані відображаються два рядки: "Example1_2" і "Example1_3". Перша з них відзначена як обрана за замовчуванням (вказано параметр SELECTED). Для доступу до третього рядка ("Example1_1") необхідно прокрутити. В даному прикладі, оскільки параметр MULTIPLE не заданий, можна відзначити тільки один рядок з заданого набору. Вказавши параметр MULTIPLE, можна відзначити будь-яку кількість рядків елемента SELECT.

Кожний зазначений рядок передається на сервер у вигляді пари "ім'я = значення". У якості імені береться ім'я набору, тобто "Number". У якості значення береться значення параметра VALUE, а якщо він не заданий – текст після тега <OPTION>. Наприклад, для другого рядка на Web-сервер передається пара "number = second", а для третього рядка, якщо він буде обраний, – "number = Example1_3". У разі передачі на сервер відразу декількох рядків, для кожного з них вказується одне і те ж ім'я (в даному прикладі – number).

Як приклад створення форми наведено HTML-документ, який реалізує різні види елементів форми. На екран виводиться Web-сторінка, що містить наступні елементи форми (в дужках вказані значення елементів, задані напочатку в HTML-документі):

- поле введення однорядковій текстової інформації (рядок тексту);
- поле введення багаторядкової текстової інформації (багаторядкова
- інформація);
- поле введення пароля;
- незалежні перемикачі (box1 box3);
- залежні перемикачі (on2);
- поле вибору одного або декількох елементів списку (second third);
- кнопка "СБРОС";
- кнопка "ПЕРЕДАЧА".

Значення всіх елементів форми (крім кнопок "СБРОС" і "ПЕРЕДАЧА") можуть бути змінені пользователем. При натисканні кнопки "СБРОС" відновлюються вихідні значення елементів. При натисканні кнопки "ПЕРЕДАЧА" задані значення передаються на сервер. Але оскільки в задачі даної лабораторної роботи не входить передача даних на сервер (це буде здійснено в лабораторній роботі 9), то для перевірки роботи кнопки "ПЕРЕДАЧА" в якості значення параметра ACTION вказується ні URL-адрес

серверної програми, а шлях до файлу, який містить HTML-документ – prim2-1.htm. У цьому випадку буде завантажена Web-сторінка ***прикладу 8.1*** лабораторної роботи 2.

Приклад 8.1

```
      <HTML>
      <HEAD>
<TITLE> Форма на Web-странице </TITLE>
      <LINK REL=stylesheet HREF="my_d.css">
      <STYLE>
      .form{background:#a0ffff;color:#004000;font-size:6mm}
      .kn{background:#e0e0ff; border:outset 4 blue; font-
size:4.5mm}
      </STYLE>
      </HEAD>
      <BODY>
      <H1 Создание формы</H1>
      <FORM METHOD=get ACTION="prim2-1.htm">
      <TABLE ALIGN=center STYLE="color:#000080; font-
size:6mm">
      <TR> <TD VALIGN=top> Поле ввода однострочного текста
(text)
      <TD><INPUT TYPE=text SIZE=15 CLASS="form" NAME="txt1"
      VALUE="Строка текста">
      <TR><TD VALIGN=top> Поле ввода пароля (password)
      <TD><INPUT TYPE=password SIZE=15 MAXLENGTH=3
      CLASS= "form" NAME="pass">
      <TR><TD VALIGN=top> Поле ввода многострочного текста
(textarea)
      <TD><TEXTAREA NAME="txt2" ROWS=3 CLASS= "form">
      Многострочная информация</TEXTAREA>
      <TR><TD VALIGN=top> Независимые переключатели (checkbox)
      <TD>
      <INPUT TYPE=checkbox NAME="ch1" VALUE= "box1" CHECKED>1
      <INPUT TYPE=checkbox NAME="ch2" VALUE="box2" > 2 <BR>
      <INPUT TYPE=checkbox NAME="ch3" VALUE="box3" CHECKED>3
<BR>
      <TR> <TD VALIGN=top> Зависимые переключатели (radio)
      <TD><INPUT TYPE=radio NAME="rad" VALUE="on1"> Первый
<BR>
```

```
      <INPUT TYPE=radio NAME="rad" VALUE="on2" CHECKED> Второй
<BR>
      <INPUT TYPE=radio NAME="rad" VALUE="on3"> Третий <BR>
      <TR><TD VALIGN=top> Выбор из списка (SELECT)
      <TD><SELECT NAME="sel" SIZE=1 MULTIPLE CLASS="form">
      <OPTION VALUE="first"> Пример №1
      <OPTION SELECTED VALUE="second"> Пример №2
      <OPTION SELECTED VALUE="third"> Пример №3
      <OPTION VALUE="fourth"> Пример №4
      </SELECT>
      </TABLE><BR>
      <INPUT TYPE=reset VALUE= "СБРОС" CLASS="kn">  
      <INPUT TYPE=submit VALUE="ПЕРЕДАЧА" CLASS="kn">
      </FORM>
      </BODY>
      </HTML>
```

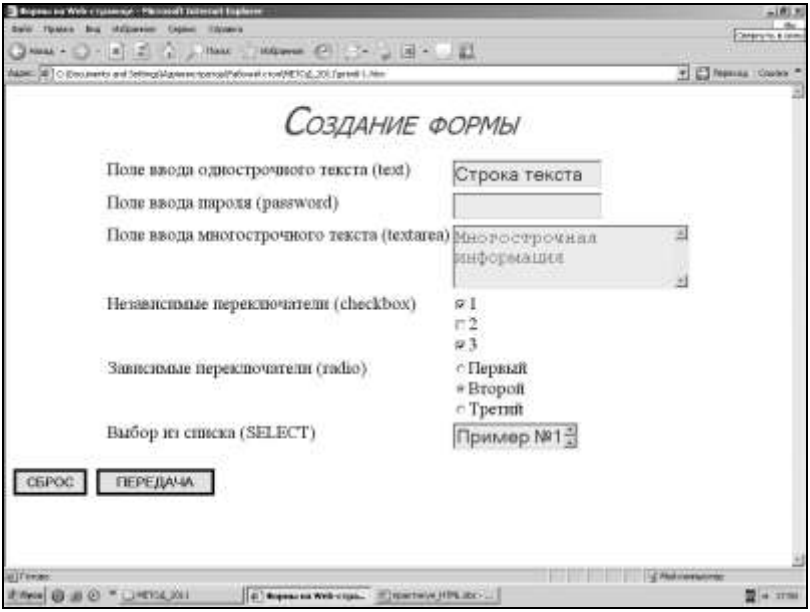


Рисунок 8.1 – Результат прикладу 8.1 на екрані

Індивідуальні завдання

Розробити HTML-документ, який задає у вигляді форми на Web-сторінці АНКЕТУ для визначення серед населення попиту і пропозицій на ту чи іншу продукцію, котра випускається тією чи іншою фірмою (продукція і фірма вибираються самостійно). Анкета також повинна містити дані про самого анкетованого. У формі необхідно вказати такі органи керування:

- поля введення однорядкової текстової інформації (не менш двох);
- поля введення багаторядкової текстової інформації (не менш двох);
- поле введення пароля;
- залежні перемикачі;
- незалежні перемикачі;
- списки рядків з вибором тільки одного рядка або відразу декількох рядків;
- кнопку скидання;
- кнопку або зображення для передачі даних.

ЛАБОРАТОРНА РОБОТА 9
ОСНОВИ МОВИ JAVASCRIPT

Мета: розгляд способів включення фрагментів JavaScript у HTML-документ, типів даних і методів роботи з екраном.

Загальні положення

До недоліків мови HTML слід віднести, що Web-сторінки, які створені за його допомогою, є статичними. Всі зміни, які відбуваються на екрані, здійснює браузер, завантажуючи, наприклад, нову Web-сторінку. Щоб зробити зображення Web-сторінки живим і динамічним, треба дати можливість користувачу інформацію прямо взаємодіяти з її змістом. Для цього необхідно до мови розмітки HTML і каскадних листків стилю CSS додати такі програмні засоби, які б дозволили на боці клієнта без участі сервера вирішити це задання.

У 1995 році фірмою Sun була розроблена мова Java, яка підтримує кросплатформенні програми. Це стало можливим за рахунок того, що додатки компілюються у якийсь проміжний код, що зветься байт-кодом Java, і виконуються на комп'ютерах клієнтів за допомогою віртуальної машини, яка розуміє байт-код (по суті інтерпретатора байт-коду).

Невеликі програми Java, призначені для вбудовування у HTML-документ, стали називатися аплетами (applets). Вони дозволили створювати по-справжньому динамічні Web-сторінки. Проте їх недоліком є те, що будь-яка зміна у програмі (наприклад, у результаті помилки), вимагає компіляції вихідного коду.

Рік потому співробітником фірми Netscape Communications Бренданом Ейхом (Brendan Eich) була розроблена на базі Java мова JavaScript (спочатку називалась LiveScript), яка була інтерпретованою мовою (скриптом) і, отже, не мала цього недоліку. Першим браузером, який підтримував JavaScript, став браузер Netscape Navigator версії 2.0.

Незабаром переваги розробки Web-сторінок з використанням JavaScript усвідомили і інші фірми, у першу чергу фірма Microsoft, яка розробила для свого браузера Microsoft Internet Explorer 3.0 версію мови

JavaScript (JScript), яка, на жаль, не повністю збігалася з версією фірми Netscape.

З метою уніфікації версій мови JavaScript у 1997 році організацією ECMA (Europen Computer Manufacturing Association) був випущений стандарт на мову JavaScript, який називався ECMAScript (його специфікацію можна знайти за адресою: http://www.esma1/STAND/ESMA-262.HTM). Хоча після цього розбіжності у версіях JavaScript і були зменшені, але не подолані повністю.

9.1. Вбудовування JavaScript у HTML-документ

Існує кілька способів включення фрагментів програми (кодів) на JavaScript у HTML-документ:

- включення кодів JavaScript між тегами <SCRIPT> і </SCRIPT>;
- підключення зовнішнього файла з кодами JavaScript за допомогою тега <SCRIPT>;
- використання кодів JavaScript безпосередньо у тегах HTML при заданні обробників подій;
- використання псевдопротокола JavaScript: URL у тегах HTML.
- Зараз розглянемо перші два способи вбудовування JavaScript у HTML-документ. З іншими познайомимося в міру вивчення мови.

9.1.1. Включення JavaScript між тегами <SCRIPT> і </SCRIPT>

Для включення фрагментів програми на JavaScript (оголошень змінних, описів функцій, операторів, викликів функцій та ін.) зазвичай використовується такий шаблон:

```
<SCRIPT TYPE="text/javascript" >
<!-- Маскування сценарію
.   .   .
Фрагменти сценарію JavaScript
.   .   .
// -->
</SCRIPT>
```

Параметр TYPE задає скрипт (мову програмування), який використовується в тезі-контейнері <SCRIPT>. Крім цього, відповідно до рекомендацій організації W3C за допомогою тега <META> повинен бути заданий скрипт, який приймає браузером за замовчуванням. Тому, якщо при розробці сценарію використовується мова JavaScript, то параметр LANGUAGE можна не вказувати.

Використання HTML-коментарю (<!-- -->) у шаблоні призначено для маскування JavaScript-сценарію для тих браузерів, які його не підтримують (версії Netscape Navigator до 2.0 і Microsoft Internet Explorer до 3.0). Інакше Web-сторінка буде відтворена неправильно. При цьому теги <SCRIPT> і </SCRIPT> цими браузерами будуть пропущені, оскільки браузери ігнорують теги, які не можуть розпізнати.

Браузери, які дозволяють інтерпретувати JavaScript, HTML-коментар розпізнають інакше: його початок (<!--) вважається однорядковим коментарем, а закінчення (-->) ігнорується.

Для браузерів, які не підтримують JavaScript та інші скрипти, можна скористатися тегом-контейнером <NOSCRIPT> </NOSCRIPT> для того, щоб повідомити про це користувачеві:

```
<NOSCRIPT>
<B> Ця Web-сторінка містить фрагменти JavaScript.
Використовуйте браузер, який підтримує цю мову. </B>
</NOSCRIPT>
```

Хоча ці браузери зараз практично не використовуються, тим не менше рекомендується використовувати запропонований шаблон вбудовування програм під час роботи зі сценаріями на JavaScript. У ***прикладі 9.1*** реалізований саме цей підхід. Хоча надалі з метою більш компактного викладу програм на JavaScript буде використовуватися скорочений варіант:

```
<SCRIPT>
...
Фрагменти сценарію JavaScript
...
</SCRIPT>
```

9.1.2. Підключення зовнішнього файла з JavaScript

Для підключення зовнішнього файла з JavaScript-кодами використовуються ті ж самі теги <SCRIPT> і </SCRIPT>, але на відміну від внутрішнього вбудовування тег <SCRIPT> містить параметр SRC, який задає URL-адресу зовнішнього файла з JavaScript.

Зазначимо, що у випадку використання параметра SRC у тегу <SCRIPT> тег, що закриває </SCRIPT> необхідний, хоча дані, які знаходяться між цими тегами ігноруються. Тобто не можна поєднати в одному тегу-контейнері <SCRIPT>…</SCRIPT> відразу і внутрішнє і зовнішнє підключення кодів JavaScript.

9.2. Особливості JavaScript

9.2.1. Структура програми і лексика мови

JavaScript як мова програмування не може використовуватися самостійно. Вона не має цілісної структури. Її окремі фрагменти, які включають описи змінних, функцій, класів і оператори, можуть бути застосовні тільки шляхом їх вбудовування у HTML-документ. Такі фрагменти будемо називати програмами чи кодами мови JavaScript.

JavaScript є чутливою до регістру мовою, тобто всі ключові слова, імена змінних, функцій і класів повинні бути написані з урахуванням регістру. У той же час HTML є нечутливим до регістру, і тому звернення до функцій JavaScript з тегів HTML можуть не дотримуватися регістру. Крім того, у браузері MS Internet Explorer усі об'єкти, їх методи і властивості, які не є частиною ядра JavaScript, а були додані як пов'язані з браузером, є також нечутливими до регістру.

JavaScript ігнорує всі пробіли, символи табуляції та переходи на новий рядок, якщо тільки вони знаходяться не всередині текстового рядка.

Усі прості оператори повинні закінчуватися символом ";" (крапкою з комою). Він служить для відділення операторів один від одного, але може бути пропущений, якщо оператори знаходяться кожен в окремому рядку. Проте не рекомендується пропускати символ "крапка з комою".

JavaScript підтримує коментарі двох видів: як у C, так і у C^{++}. Усі символи, які йдуть після комбінації //, ігноруються до кінця рядка. Коментарем вважається також будь-яка послідовність символів між /* та */. Прийнятий у HTML формат опису коментаря між <!-- та --> у JavaScript

підтримується наступним чином: усі символи до кінця рядка після <!-- ігноруються, --> не є коментарем.

Що стосується операторів, то враховуючи, що JavaScript – це С-подібна мова програмування, будуть описані тільки ті оператори JavaScript, яких немає у С.

9.2.2. Типи даних

JavaScript підтримує наступні типи даних:

- строковий;
- цілочисельний;
- з плаваючою комою;
- логічний;
- null.

Рядок задає звичайні рядки тексту. Ці рядки мають братися в одиночні (') або подвійні (") лапки. При цьому текст, обмежений подвійними лапками, може містити одиночні і навпаки.

Для включення у рядок символів, які виконують у JavaScript службові функції (', " та \), а також символів, пов'язаних з роботою клавіатури, використовуються набори символів, що іменуються ESC-послідовностями. У таблиці 9.1 наведено співвідношення між ESC-послідовностями та символами, які вставляються у рядок замість них.

Таблиця 9.1 – Співвідношення між ESC-послідовностями та символами, які вставляються у рядок замість них

ESC-послідовність	**Символ**
\'	'
\"	"
\\	\
\b	Backspace
\n	Новий рядок
\r	Повернення каретки
\t	Горизонтальна табуляція

Для об'єднання декількох рядків (виконання операції конкатенації) використовується оператор +.

Цілочисельний тип задає цілі числа від -10^{308} до 10^{308}. Вони можуть бути десятковими, вісімковими або шістьнадцятковими. Вісімкови позначаються лідируючим 0, за яким слідують цифри 0-7. Шістьнадцяткови позначаються знаками 0x або 0X, які розміщені на початоку, і складаються з цифр 0-9 і букв A-F або a-f.

Тип з плаваючою точкою задає речові числа, які містять цілу і дробову частини, розділені крапкою. Точка ставиться навіть у тих випадках, коли дробової частини немає. Числа з плаваючою точкою можуть бути записані також в експоненційній формі {мантиса}E{порядок}. Дійсні числа можуть бути тільки десятковими.

Логічний тип задає величини, які можуть набувати тільки двох значень вигляду "так/ні", яким відповідають ключові слова JavaScript true і false ("істина", "брехня").

Тип даних, який характеризує відсутність даних, позначається ключовим словом null.

Літералами називаються дані будь-якого типу, записані відповідно доправил мови програмування, у даному випадку, JavaScript. Наприклад, літералами є наступні дані, відповідно рядкового, цілочисельного і типу з плаваючою крапкою:

- "It's a cat";
- 0xA5;
- 1.8E-1.

9.2.3. Оголошення змінних

Імена змінних JavaScript, так само як у мовах Pascal і C, можуть містити латинські літери, цифри і знаки підкреслення та починатися або з літери, або зі знака підкреслення. Але при оголошенні змінної, на відміну від цих мов, тип змінної не вказується. Змінна набуває тип тих даних, які їй присвоюються. Тому в різний час змінна може мати різний тип даних.

Оголосити змінну можна одним з наступних способів:

- за допомогою оператора var, наприклад, `var a, b`;
- за допомогою оператора var з присвоєнням значення змінної, наприклад, `var a, b="It's a cat"`;

- при присвоєнні значення змінної, наприклад, `a=1.8E-1; C=042;`

Змінні, оголошення яких у програмі здійснюється поза описами функцій, називаються глобальними. Їх областю дії є вся програма, включаючи області опису функцій. Для цих змінних усі вищенаведені види оголошень змінних є рівноправними.

Для локальних змінних, тобто таких, які оголошені всередині опису функцій, використання при оголошенні змінної оператора var приведе до того, що змінна стає недоступною для всієї програми, за винятком області опису функції. Якщо локальна змінна оголошена шляхом присвоєння їй значення без використання оператора var, вона буде доступна для всієї програми.

Літерали, змінні або функції, які з'єднані знаками операцій, є виразом у мові JavaScript, наприклад, a-C +6.13.

Тип змінної (або в загальному випадку висловлення) можна визначити за допомогою оператора typeof (допускається і його форма у вигляді функції typeof ()), яке поверне такы значення:

- number – для змінних цілого і дійсного типів;
- string – для змінної строкового типу;
- boolean – для змінної логічного типу;
- null – для змінної, яка не має значення чи не оголошена.

Якщо змінна типу string на початку рядка містить символи, які становлять ціле або речове число, наприклад, a=0xAAWWW, або b='1.43e1cats', то за допомогою функцій parseInt() або parseFloat можна перетворити тип змінної відповідно у цілий або речовинний тип. Наприклад, функція parseFloat (b) поверне значення 14.

9.2.4. Вбудовані функції

JavaScript має свої власні функції, які можна використовувати в програмі. Вони називаються вбудованими, оскільки оголошені і реалізовані усередині інтерпретатора мови. До них належать такі функції:

- escape(рядок) – повертає рядок (в форматі Unicode). Всі пробіли, пунктуація і будь-які не-ASCII символи в ній закодовані і виглядають як %xx, де xx еквівалентно шестнадцатиричі числа, які позначають символ. Наприклад, пробіли будуть повернуті як "% 20". Символи, числові значення яких найбільше 255, будуть представлені в форматі% uxxxx;

- eval(рядок) – якщо аргументом є вираз, то функція повертає обчислене значення виразу, якщо аргументом є оператор або послідовність операторів, то функція виконує ці оператори;
- isFinite(вираз) – функція повертає true, якщо аргумент є кінцевим числом (уявленим в JavaScript), або рядком, що є таким числом, узятим в лапки, наприклад, "125" або "1.2E3" і повертає false, якщо:
  - значення аргументу функції перевищує найбільш допустиме в JavaScript число (1.7976931348623158E + 308), в цьому випадку при виведенні числа на екран вказується не число, а властивість Infinity (нескінченність);
  - значення аргументу функції менш найменшого допустимого в JavaScript числа (□1.7976931348623158E + 308), в цьому випадку при виведенні числа на екран вказується властивість -Infinity (до нескінченності);
  - значення аргументу функції не є числом, тобто має властивість NaN (Not a Number – не число);
- isNaN(вираз) – повертає true, якщо аргумент не є числом і false в іншому випадку;
- parseFloat(рядок) – визначає, чи є перший символ рядка цифрою. Якщо це так, то здійснює синтаксичний розбір рядка до досягнення закінчення числа і повертає це число з плаваючою точкою як число, а не як рядок. Якщо ж перший символ не є цифрою, функція повертає NaN.

Можна відзначити, що функція повертає тільки одне число і допускає в аргументі попередні та наступні пробіли:

- parseInt(рядок[, основа]) – повертає ціле число шляхом синтаксичного аналізу рядка, при цьому необов'язковий другий параметр використовується для зазначення підстави системи числення, наприклад, значення 16 вказує, що число в рядку повинно бути перетворено з шестнадцатиричного числа в десяткове.

Якщо другий параметр пропущено, то приймається:

- якщо рядок починається з "0x", то основа дорівнює 16;
- якщо рядок починається з "0", то основа дорівнює 8;
- якщо рядок починається з будь-якого іншого значення, то основа дорівнює 10;
- якщо перетворення не може бути виконано, то функція повертає NaN.

9.3. Робота з екраном на JavaScript

Мова JavaScript є об'єктно-орієнтованою мовою. Об'єктом самого високого рівня у ньому є об'єкт window, який має кілька методів (функцій) для роботи з екраном. У цій лабораторній роботі розглянемо наступні три методи:

- alert();
- confirm();
- prompt().

9.3.1. Метод alert()

При виконанні функції alert() на екрані браузера створюється стандартне вікно попередження, у якому виводяться дані, які задані як аргумент цієї функції. Після натискання кнопки "OK" вікно методу alert() закривається.

У ***прикладі 9.1*** показано застосування цієї функції для виведення на екран повідомлення "Значення змінних:", повернення каретки на новий рядок і виведення значень змінних (y, a і _b), оголошених у програмі:

Приклад 9.1

```
<SCRIPT TYPE="text/javascript" >
<!-- Маскирование сценария
var y,a=0xA5;
_b=2.8e-1;
alert ("Значения переменных:\n"+y+","+a+","+_b);
// -->
</SCRIPT>
```

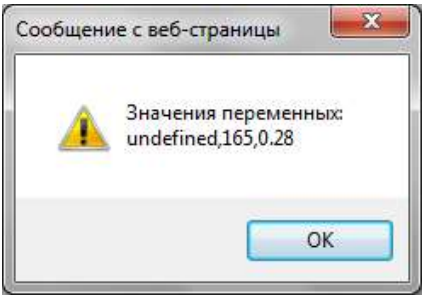


Рисунок 9.1 – Результат прикладу 9.1 на екрані

При запуску ***прикладу 9.1*** здійснюється виконання наведеного фрагмента програми, у результаті якого на екран будуть виведені такі значення змінних: undefined, 165, 0.28. Тобто змінна y, якій нічого не було присвоєно, набуває значення undefined, значення цілої змінної a перетворено у десяткову форму, а значення дійсної змінної _b – у неекспоненціальну форму.

9.3.2. Метод confirm()

Функція confirm() використовується для організації інтерактивної взаємодії з користувачем. При виконанні цієї функції, аргументом якої є рядок, що містить питання, на екрані створюється стандартне запитуване вікно, в якому вказується це питання. Крім того, вікно містить дві кнопки: "OK" і "Скасувати" ("Cansel"). Якщо користувач відповідає на питання позитивно, тобто "так", він натискає кнопку "OK". У цьому випадку функція confirm() набуває значення true. Якщо на питання дається негативна відповідь, то необхідно натиснути кнопку "Скасувати". При такій відповіді функція confirm() набуває значення false.

У ***прикладі 9.2*** показано застосування цієї функції для визначення зацікавленості користувачів у вивченні мови JavaScript.

9.3.3. Метод prompt()

Функція prompt() призначена для введення користувачем текстової інформації (якщо вводяться цифри, вони можуть бути перетворені у число за допомогою функцій parseInt() або parseFloat()).

Функція prompt() може мати один або два рядкових аргументаи При виконанні функції на екрані формується вікно, яке має такі елементи: поле запиту сценарію, куди виводиться значення першого (або єдиного) аргументу функції; поле введення інформації від користувача і дві кнопки – "OK" і "Скасувати".

Якщо вказано один аргумент функції, наприклад, prompt ("Введіть ім'я"), то у полі введення з'явиться значення undefined. Тому такий варіант задання аргументів не рекомендується застосовувати.

При використанні двох аргументів у полі введення виводиться значення другого аргументу, наприклад, при виконанні функції prompt ("Введіть ім'я", "введіть тут"), у полі введення з'явиться рядок "введіть тут". Якщо вказати, як у ***прикладі 9.2***, пустий рядок, то у полі введення нічого виведено не буде.

Після появи на екрані вікна користувач заповнює поле введення і натискає кнопку "OK". У цьому випадку функція prompt() набуває значення введених даних рядкового типу. Якщо поле введення залишилося незаповненим або користувач натиснув кнопку "Скасувати", функція набуває значення null (для оператора if значення undefined і null сприймаються як false).

У ***прикладі 9.2*** показано застосування цієї функції для введення імені користувача, яке буде занесене у змінну user_name. Після цього здійснюється перевірка цієї змінної на наявність значення з висновком відповідного повідомлення на екран.

Приклад 9.2

```
<SCRIPT>
if (confirm("Ви хочете вивчати JavaScript?"))
alert("Дуже добре!");
else alert("Але ж може знадобиться!");
user_name=prompt("Введіть ім'я","");
if (user_name) alert("Введено ім'я:"+ user_name);
else alert("Ім'я не введено")
</SCRIPT>
```

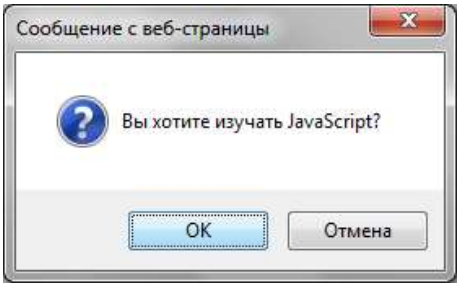


Рисунок 9.2 – Результат прикладу 9.2 на екрані

Індивідуальні завдання

Розробити на JavaScript з використанням методів alert(), confirm() і prompt() програму тестування знань студентів з HTML і CSS, яка повинна забезпечити виконання наступних дій:

- введення студентами назви групи і свого прізвища;
- виведення на екран не менше чотирьох питань із двома варіантами відповідей кожне ("так" і "ні") за темою з таблиці 9.2 згідно з варіантом (номера студента в журналі);
- визначення на підставі відповідей студента кількості отриманих ним балів;
- висновок результатів тестування на екран з вказаними прізвищем студента, його групи і отриманої оцінки.

Таблиця 9.2 – Варіанти індивідуальних завдань

№	Тема перевірки знань
1	2
1	Форматування тексту засобами HTML.
2	Форматування тексту засобами CSS.
3	Вивід списків на Web-сторінці засобами HTML і CSS.

Продовження табл. 9.2

1	2
4	Вивід таблиць на Web-сторінці.
5	Вивід зображень на Web-сторінці.
6	Створення фону на Web-сторінці.
7	Вивід відеофрагментів на Web-сторінці.
8	Використання посилань на Web-сторінці.
9	Використання фреймів на Web-сторінці.
10	Використання плаваючих фреймів на Web-сторінці.
11	Створення форми на Web-сторінці. Поле введення тексту.
12	Створення форми на Web-сторінці. Залежні перемикачі.
13	Створення форми на Web-сторінці. Незалежні перемикачі.
14	Створення форми на Web-сторінці. Тег SELECT.
15	Використання інтерфейсу CGI для обробки форми.
16	Вирівнювання тексту
17	Задання розміру символів
18	Форматування HTML-документа
19	Структура URL-адреса
20	Задання кольору на Web-сторінці

ЛАБОРАТОРНА РОБОТА 10
РОБОТА З МАСИВАМИ НА JAVASCRIPT

Мета: вивчення способів і функцій роботи з індексними і асоціативними масивами.

10.1. Індексні масиви

10.1.1. Оголошення масиву

Індексними називаються такі масиви, доступ до елементів яких здійснюється за номерами (індексами) цих елементів. Такими є масиви у мовах Pascal та C. Індексні масиви JavaScript відрізняються від них тим, що можуть містити елементи різних типів. Нумерація елементів індексного масиву починається з індексу 0.

Масиви у JavaScript оголошуються як об'єкти за допомогою конструктора new Array(). Якщо конструктору не передається ні один параметр, то створюється масив нульової довжини. Наприклад, таким чином створюється масив arr:

```
arr=new Array();
```

Якщо конструктору передається один параметр, то створюється масив, який має розмір, рівний значенню цього параметра:

```
arr=new Array(5);
```

У даному випадку число елементів масиву arr є 5.

10.1.2. Задання елементів масиву

Існує кілька способів задання елементів масиву:

- поелементне задання;
- за допомогою конструктора при оголошенні;
- за допомогою літералу.

При поелементному заданні для кожного елемента масиву вказується його індекс і значення:

```
arr[0]=0x7D;
arr[2]=2.8e-1;
arr[3]="It's cat";
```

Для перевірки правильності задання елементів масиву виведемо їх значення на екран. Для цього скористаємося функцією alert(). Якщо як параметр цієї функції вказати ім'я масиву

```
alert(arr),
```

то отримаємо висновок значень масиву в стандартній формі. Клацнувши по оператору виклику функції, отримаємо ці значення на екрані у вигляді таких значень, розділених комою: 125,, 0,28, It's cat. Видно, що число з 16-кової форми було перетворено у десяткову, дійсне число перетворено у неекспоненціальну форму, а дві коми підряд означають, що не заданий елемент масиву arr[1].

Можна вивести значення елементів масиву у довільній формі, наприклад, так:

```
a="";
for (j=0;j<arr.length;j++) a+="arr["+j+"]="+arr[j]+"\n";
alert(a).
```

Зазначимо, що кількість елементів масиву визначено за допомогою властивості length. Натиснувши на функції alert(a), отримаємо вивід елементів масиву і їх значень на екран вже в іншій формі. Зазначимо, що замість коми для незаданого елемента масиву вказується значення undefined.

Можна задати масив відразу при його оголошенні, вказавши конструктору за параметри значення елементів масиву:

```
arr=new Array(0x7D,2.8E-1,"It's cat");
```

Цей масив буде трохи відрізнятися від масиву, отриманого раніше, тим, що не буде містити невизначеного елемента, оскільки дозволяє задавати елементи масиву тільки з індексу 0 і підряд.

Задання масиву за допомогою літерала здійснюється наступним чином:

```
arr=[0xA5,,1.25E4,"It's cat"];
```

При використанні літерала отримаємо такий самий масив, що і в першому випадку.

10.2. Функції для роботи з індексними масивами

У таблиці 10.1 наведені функції (методи), які можуть бути використані при роботі з індексними масивами.

Таблиця 10.1 – Функції (методи) для роботи з індексними масивами

Метод	Опис
concat(список елементів, які додаються)	Повертає масив, отриманий після об'єднання елементів поточного масиву і елементів, перерахованих у списку. Список може містити інші масиви.
join(роздільник)	Повертає рядок, отриманий після злиття, через роздільник усіх елементів масиву.
pop()	Видаляє останній елемент масиву і повертає його. Якщо масив пустий, повертається undefined.
push(список елементів, які додаються)	Додає у масив елементи, перераховані у списку, і повертає нову довжину масиву. Список може містити інші масиви.
reverse()	Повертає масив, порядок елементів якого змінено на протилежний.
shift()	Видаляє перший елемент масиву і повертає його.
slice(індекс першого елемента, індекс останнього елемента)	Повертає масив, утворений з елементів поточного масиву, від першого елемента включно до останнього елемента включно.

Продовження таблиці 10.1

Метод	Опис
sort([функція сортування])	Повертає масив, заповнений відсортованими елементами поточного масиву. Функція сортування повинна приймати два параметри і повертати одне з наступних значень: 1, якщо перший більше другого; -1, якщо другий більше першого та 0, якщо вони однакові. Якщо функція сортування опущена, виконується символьне сортування.
splice(індекс першого елемента, кількість елементів, які видаляються, [список елементів, які додаються, і розділені комами])	Видаляє з масиву задану кількість елементів і вставляє на їх місце нові зі списку, якщо вони визначені. Повертає масив, який складається з віддалених елементів.
toSource()	Повертає рядок, який являє собою вихідний код масиву.
toString()	Аналогічний методу toSource().
unshift(список елементів, які додаються)	Повертає масив, отриманий після об'єднання елементів поточного масиву і елементів, перерахованих у списку. Причому елементи вставляються у початок поточного масиву. Список може містити інші масиви.

У ***прикладі 10.1*** показано використання методу concat() для модифікації елементів масиву.

10.3. Асоціативні масиви

Асоціативними є такі масиви, доступ до елементів яких здійснюється не за їх індексами, як у індексних масивах, а за їх ключами, за які використовуються рядкові літерали. Ключі, які складаються тільки з цифр, є індексами.

Оголошення асоціативних масивів здійснюється, так само як і індексних масивів, за допомогою конструктора new Array():

```
arr2=new Array();
```

Однак для задання значень елементів асоціативного масиву можна використовувати тільки спосіб поелементного задання ключів і значень для кожного елемента масиву.

Асоціативні масиви можуть містити як асоціативні, так і індексні елементи. Як приклад буде задано саме такий масив arr_as:

```
arr_as["name"]="Smith";
arr_as["1"]=0x7D;
arr_as[2]=2.8e-1;
arr_as["\\"]="ABC";
```

При виведенні значень елементів цього масиву на екран описаним вище способом

```
alert(arr_as);
```

виявляється, що на екран виводяться тільки індексні елементи масиву (,125,0.28). У цьому можна переконатися, клацнувши по функції виведення. Те ж відбудеться, якщо спробувати вивести асоціативні елементи масиву на екран за допомогою оператора циклу for.

Щоб отримати доступ до всіх елементів асоціативного масиву arr_as:, необхідно скористатися оператором for in, якого немає у мові С (мабуть тому, що в ньому немає асоціативних масивів):

```
a="";
for(j in arr_as) a+=j+"=>"+arr_as[j]+"\n";
alert(a);
```

Натиснувши на функцію виведення, побачимо, що у цьому випадку на екран виведені ключі і значення всіх елементів масиву.

Оператор for in можна використовувати не тільки для доступу до елементів асоціативного масиву, але і для отримання властивостей (ключів) і значень об'єктів мови JavaScript.

У ***прикладі 10.1*** показано задання елементів індексного масиву літеральним способом, модифікація масиву за допомогою методу concat(), визначення елемента цього масиву, значенням якого є рядок, який стоїть раніше за інших за алфавітом, тобто має мінімальний ASCII-код, а також визначення властивостей об'єкта navigator.

Приклад 10.1

```
<SCRIPT>
arr=[0x7D,"one",2.8e-1,"It's cat",'twenty five'];
alert(arr);
arr_mod=arr.concat(99,"Tiger",0123);
alert(arr_mod);
min='zzz';
for (j=0;j<arr_mod.length;j++)
if (typeof (arr_mod[j])=='string') if(arr_mod[j]<min)
{Min_j=j;min=arr_mod[j];}
alert("arr_mod["+min_j+"]="+min);
</SCRIPT>
```

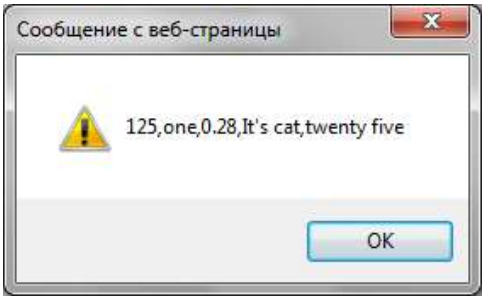


Рисунок 10.1 - Результат прикладу 10.1 на екрані

Зверніть увагу, що як початкове мінімальне значення елемента обраний рядок 'zzz', тобто передбачається, що елементи масиву не будуть мати рядки з великим ASCII-значенням.

Індивідуальні завдання

Мовою JavaScript розробити програму, яка виконує згідно з вказаними у таблиці 10.2 індивідуальними параметрами такі дії:

- створити індексний масив, який містить не менше 10 елементів, з яких 5 чисел (цілих і речових) і 5 рядків за допомогою способу, заданого у колонці 2;
- над елементами створеного масиву виконати функцію, зазначену в колонці 3;
- визначити тип елементів модифікованого масиву і для чисел або для рядків виконати операцію, задану в колонці 4;
- створити асоціативний масив з не менш 6 елементів;
- над ключами або значеннями виконати операцію, зазначену в колонці 6;
- після виконання кожного з перерахованих вище пунктів вивести зміст масиву на екран.

Таблиця 10.2 – Варіанти індивідуальних завдань

№	Створення масиву	Функція	Операція	
			індексний масив	асоціативний масив
1	2	3	4	5
1	Поелементне	join()	знайти максимальне число	знайти ключ, перший за алфавітом
2	При оголошенні	pop()	знайти мінімальне число	знайти ключ, останній за алфавітом
3	Літеральне	push()	знайти середнє арифметичне чисел	знайти середню довжину ключа
4	Поелементне	reverse()	знайти середню довжину	знайти ключ мінімальної довжини
5	При оголошенні	shift()	знайти рядок, останній за алфавітом	знайти ключ максимальної довжини
6	Літеральне	slice()	знайти рядок максимальної довжини	знайти ключ, перший за алфавітом
7	Оелементне	splice()	знайти рядок мінімальної довжини	знайти ключ, останній за алфавітом
8	При оголошенні	onshift()	знайти максимальне число	знайти середню довжину ключа
9	Літеральне	join()	знайти мінімальне число	знайти ключ максимальної довжини
10	Поелементне	pop()	знайти середнє арифметичне чисел	знайти ключ мінімальної довжини
11	При оголошенні	push()	знайти середню довжину рядків	знайти ключ, перший за алфавітом
12	Літеральне	reverse()	знайти рядок, останній за алфавітом	знайти середню довжину ключа
13	Поелементне	shift()	знайти рядок максимальної довжини	знайти ключ, останній за алфавітом
14	При оголошенні	slice()	знайти рядок мінімальної довжини	знайти ключ, перший за алфавітом
15	Літеральне	splice()	знайти максимальне число	знайти ключ, останній за алфавітом

Продовження табл. 10.2

1	2	3	4	5
16	Поелементне	onshift()	знайти мінімальне число	знайти середню довжину ключа
17	При оголошенні	join()	знайти середнє арифметичне чисел	знайти ключ, перший за алфавітом
18	Літеральне	pop()	знайти середню довжину рядків	знайти ключ, останній за алфавітом
19	Поелементне	push()	знайти рядок, останній за алфавітом	знайти ключ максимальної довжини
20	При оголошенні	reverse()	знайти рядок максимальної довжини	знайти ключ, останній за алфавітом

ЛАБОРАТОРНА РОБОТА 11
ОБРОБКА ПОДІЙ І ЗМІНА ВЛАСТИВОСТЕЙ ОБ'ЄКТІВ НА WEB-СТОРІНЦІ

Мета: вивчення засобів і можливостей мови JavaScript для обробки подій на Web-сторінці з метою визначення та зміни властивостей її об'єктів

11.1. Обробка подій

11.1.1. Події Web-сторінки

Усі зміни, які відбуваються на Web-сторінці, пов'язані з роботою браузера або маніпуляціями користувача з клавішами мишки або клавіатури, називаються подіями. Для вказівки дій, які необхідно зробити у зв'язку з появою тієї чи іншої події, використовуються обробники подій. Вони вказуються у вигляді назви події з додаванням префікса on.

JavaScript підтримує наступні обробники подій, пов'язаних з роботою браузера:

- onLoad − викликається по події Load: завантаження Web-сторінки;
- onUnload − викликається по події Unload: завантаження Web-сторінки при завершенні роботи зі сторінкою або при переході на іншу сторінку.
- onBeforeunload − викликається за подією beforeunload при підготовці до вивантаження Web-сторінки по завершенню роботи зі сторінкою або при переході на іншу сторінку (тільки IE).

Подія Load часто використовується, коли необхідно виконати дії відразу після завантаження Web-сторінки до виконання обробників інших подій.

Подія Unload використовується дуже рідко, оскільки вивантаження Web-сторінки скасувати вже не можна і будь-які дії не потрібні.

Подія onbeforeunload може бути використана для підтвердження готовності покинути поточну Web-сторінку.

При роботі з мишкою JavaScript підтримує такі обробники подій (вказуються у порядку їх обробки браузером):

- onMouseover − викликається по події Mouseover: наведення вказивніка миші на об'єкт;

- onMousemove − викликається по події Mousemove: вказівник мишки поміщається на об'єкт або рухається по об'єкту;
- onMousedown − викликається по події Mousedown: натискання лівої чи правої клавіш мишки;
- onMouseup − викликається по події Mouseup: віджатіє лівої клавіші мишки;
- onClick − викликається по події Click: клацання мишкою (натисніть і віджаті лівої клавіші мишки);
- onDblclick − викликається по події Dblclick: подвійне клацання мишкою;
- onMouseout − викликається по події Mouseout: вказівник мишки знімається з об'єкту:
- onContextmenu − викликається по події Contextmenu: віджатіє правої клавіші мишки.
- onMouseout − викликається за подією mouseout при знятті покажчика мишки з об'єкта;
- onMousewheel − викликається за подією mousewheel при прокручуванні колеса мишки.

11.1.2. Завдання обробників подій

Є три моделі зазначення (реєстрації) обробника події в HTML-документі, які умовно можна назвати:

- вбудована (inline);
- традиційна (traditional);
- вдосконалена (advanced).

11.2.1 Використання вбудованої моделі

У разі використання вбудованої моделі обробник подій розміщується в тег як параметр, при цьому використовується такий вираз:

```
Оброблювач події="дії з обробки події",
```

де:

- оброблювач події − це один з перерахованих вище обробників подій;
- дії з обробки події − це вбудований фрагмент програми на JavaScript, який може включати оператори та/або виклик функції.

Наприклад, при натисканні мишкою по заголовку Web-сторінки в разі розміщення в тег <H1> обробника цієї події (зазвичай він вказується останнім параметром):

```
<H1 ID="h" onClick="outText()">Заголовок</H1>,
```

викликається функція outText ()

```
function outText() {alert('Тэг отмечен') },
```

яка виводить на екран повідомлення "Тег відмічений".
Можна для цієї мети також безпосередньо використовувати оператор:

```
<H1 ID="h" onClick="alert('Тэг отмечен')">Заголовок </H1>.
```

Оброблювач події поміщається у той тег, який повинен реагувати на дії користувача з мишкою, хоча при цьому обробником можуть бути змінені параметри та властивості будь-якого тега Web-сторінки.

Обробники подій, пов'язаних з діями браузера і подій, пов'язаних з натисканням клавіш клавіатури, розміщуються лише у тег <BODY>.

11.1.3. Об'єкт event

Часто разом з подією передаються дані, пов'язані з цією подією, наприклад, при клацанні мишки по елементу Web-сторінки передається покажчик на цей елемент. Для того, щоб програма могла отримати доступ до цих даних, у JavaScript передбачен об'єкт event, який є властивістю об'єкта window. Тоді дані, які передаються разом з подією можна отримати за допомогою формули window.event.свойство.

Наведемо деякі властивості об'єкта event (інші будуть розглянуті у наступних лабораторних роботах):

- srcElement − повертає посилання на елемент (тег) Web-сторінки, який викликав наставання події;
- fromElement − повертає посилання на елемент Web-сторінки, з якого перемістився курсор мишки при наставанні події Mouseover або Mouseout;
- toElement − повертає посилання на елемент Web-сторінки, на якє поміщений курсор мишки;

- type − повертає ім'я події;
- returnValue − задає, чи буде виконуватися дія за умовчанням (якщо true - буде, якщо false − ні);
- propertyValue − повертає ім'я атрибуту тега або стилю властивості елемента Web-сторінки, значення якого змінилося;
- button - повертає номер натиснутої кнопки мишки:
  - 0 − нічого не було натиснуто;
  - 1 − натиснута ліва кнопка;
  - 2 − натиснута права кнопка;
  - 3 − натиснуті одночасно ліва і права кнопки;
  - 4 − натиснута середня кнопка;
  - 5 − натиснуті одночасно ліва і середня кнопки;
  - 6 − натиснуті одночасно права і середня кнопки;
  - 7 − натиснуті одночасно усі кнопки.
- cancelBubble − задає, чи будє заборонена передача події батьківському елементу (якщо true − так, якщо false − ні).

Використання властивостей type та srcElement об'єкта event наведено у ***прикладі 11.2***.

11.1.4. Проходження подій

Під проходженням подій розуміється порядок, у якому подія передається для обробки від одного елемента Web-сторінки до другого.

Розглянемо ***приклад 11.1***. При його запуску на екран виводиться прямокутник, заштрихований блакитним кольором, і зображення рибок, причому останній елемент є дочірнім по відношенню до прямокутника.

Якщо клацнути мишкою по зображенню рибок, на екрані за допомогою функції alert() спочатку буде виведено повідомлення "IMAGE", потім повідомлення "OBJECT", після цього − "BODY". Це відбувається так тому, що після обробки події дочірнім елементом подія передається для обробки батьківським елементам: спочатку елементу з ідентифікатором obj (прямокутнику), потім тегу <BODY>.

Якщо небажано передавати подію батьківському елементу, можна це заборонити, задавши властивості event.cancelBubble значення true. Для цього у прикладі необхідно прибрати коментарі в рядку

```
//Event.cancelBubble=true;
```

Після чого клацанням миші по рисунку з рибками на екран буде виведено "IMAGE".

Приклад 11.1

```
<HTML>
<HEAD>
<TITLE> Прохождение событий на Web-странице </TITLE>
<SCRIPT>
function clickIMG()
{
alert("IMAGE");
//event.cancelBubble=true;
}
</SCRIPT>
</HEAD>
<BODY onClick="alert('BODY')">
<DIV  STYLE="background:#E0E0FF;  width:110;  height:110"
onClick="alert('DIV')">
<IMG SRC="fish.gif" onClick="clickIMG()">
</DIV>
</BODY>
</HTML>
```

11.2. Об'єктна модель документа

Відповідно до об'єктною модель документа DOM (Document Object Model), яка підтримується мовою JavaScript, усі теги Web-сторінки є об'єктами і поміщені в колекцію, яка називається document.all (специфікацію DOM можна отримати за адресою http://www.W3.org/DOM/). Колекція відрізняється від асоціативних масивів, які розглянуті в лабораторній роботі 10 "Робота з масивами на JavaScript" тим, що крім властивостей має також методи, тобто сама є об'єктом.

Рисунок 11.1 – Результат приклада 11.1 на екрані

Завдяки об'єктної моделі документа з'явилася можливість за допомогою JavaScript програмно отримати доступ до будь-якого об'єкту (тегу) Web-сторінки для визначення та/або зміни його властивостей за допомогою вираження об'єкт.свойство.

Як властивості об'єкта можуть бути зазначені:

- HTML-параметри тегів (записуються великими літерами);
- властивості каскадних аркушів стилів CSS;
- загальні властивості тегів, пов'язані з введенням об'єктної моделі документа:
  - tagName − найменування тега;
  - sourceIndex − порядковий номер тега у колекції document.all;
  - innerText − текстовий зміст елемента Web-сторінки, який включає текст дочірніх елементів, за винятком будь-яких тегів HTML. Якщо присвоїти елементу нове значення, весь зміст елементу буде замінено;

− outerText − текстовий зміст елемента Web-сторінки, який включає текст дочірніх елементів, за винятком будь-які теги HTML. Якщо присвоїти елементу нове значення, весь зміст елемента і сам елемент будуть замінені;

− innerHTML − весь зміст елемента: текст і теги дочірніх елементів. Якщо присвоїти цій властивості нове значення, весь зміст елементу буде замінено;

− outerHTML − весь зміст елемента: текст і теги дочірніх елементів, а також теги, які утворюють цей елемент. Якщо присвоїти цій властивості нове значення, весь зміст елемента і сам елемент будуть замінені;

− parentElement − тег-батько (для всіх тегів, крім першого у HTML-документі).

При використанні HTML-параметрів як властивостей об'єктів необхідно вказувати тільки такі параметри, які допустимі для даного тега. Наприклад, використання параметра ALIGN для тега <FONT> призведе до помилки.

Якщо у якості властивостей об'єктів використовуються властивості каскадних аркушів стилів, то, оскільки лексика CSS відрізняється від лексики JavaScript, необхідно перетворити форму властивості CSS у форму властивості об'єкта JavaScript. Для цього застосовується наступне правило: якщо властивість CSS складається з одного слова, то перетворення не потрібно; якщо властивість CSS складається з декількох слів, записаних через дефіс, то в позначенні властивості для JavaScript усі знаки дефіса прибираються, перше слово властивості пишеться з прописної літери, а друге слово і наступні слова властивості пишуться з великої літери. Наприклад, властивість CSS font-size на JavaScript буде виглядати так: fontSize, а властивість CSS list-style-image: url (URL) так: `listStyleImage="url (URL)"`.

При використанні об’єктної моделі документа доступ до усіх властивостей CSS здійснюється за допомогою об'єкту style, який має такі властивості:

- style;
- currentStyle;
- runtimeStyle.

Властивість style дозволяє здійснити доступ для визначення та/або зміни тільки ти властивості CSS, які вбудовано задані у тегу за допомогою параметра STYLE. Для цього використовується вираження style.свойство CSS, наприклад, `style.fontSize`.

Тому за допомогою властивості style не можна отримати доступ до властивостей CSS, зазначеним у тегу-контейнері <STYLE>…</STYLE>. Для цього необхідно скористатися властивістю currentStyle, яка дозволяє визначити поточні значення властивостей CSS (без їх зміни), які зазначені як за допомогою параметра STYLE, так і за допомогою тега <STYLE>, а також значення HTML-параметрів.

За допомогою властивості runtimeStyle можна змінити властивості об'єктів на екрані без зміни значень властивостей CSS параметра STYLE.

Визначення властивостей об'єктів Web-сторінки (загальних, заданих за допомогою HTML-параметрів або за допомогою властивостей CSS) наведено у ***прикладі 11.2***.

Використання об'єктної моделі документа для визначення та зміни властивостей елементів Web-сторінки наведено у ***прикладі 11.2***, у якому задаються елементи Web-сторінки і їх властивості, а також виклики обробників подій.

Обробниками подій є функції на JavaScript, що зберігаються на зовнішньому файлі "js_pr3-2.js", які підключаються до даного HTML-документу і мають таке призначення:

- функція allTags() викликається при завантаженні сторінки, визначає і виводить на екран властивості всіх елементів Web-сторінки);
- функція changeTag(obj) викликається за натисканню лівої чи правої кнопок мишки і змінює властивості елемента Web-сторінки;
- функція restoreTag(obj) викликається за віджату правої кнопки мишки і відновлює початкові властивості елемента Web-сторінки;
- функція showTags() викликається по клацанню мишки, визначає і виводить у поле статусу вікна браузера властивості зазначеного елемента Web-сторінки;
- функція changeTags() викликається за подвійним клацання мишки і змінює властивості колекції елементів Web-сторінки.

Приклад 11. 2

```
<HTML>
<HEAD>
<TITLE> Визначення і зміна властивостей елементів Web-
сторінки </TITLE>
<STYLE>
    P{color:#0000D0;font-size:8mm}
    .big_red{color:red; font:2.5cml}
</STYLE>
<SCRIPT SRC="js_pr3-2.js"></SCRIPT>
</SCRIPT>¶
</HEAD>
<BODY ID="bod"
    onLoad="allTags()"
    onClick="showTagInfo()"
    onDblclick="changeCollection()">
<H1 ID="h" ALIGN=center STYLE="color:#FF00FF;
font:12mm"> Заголовок </H1>
<H6 ID="h6"
    onMousedown="changeTag(getElementById('h6'))"
    onContextmenu="restoreTag(getElementById('h6'));
return false"> Заголовок №6 </H6>
<P ID="p1" STYLE="color:blue; font-size:4mm"> Текст
<FONT ID="f" COLOR=#C00000 SIZE=5> Ще текст> </FONT></P>
<P><B ID="bold"> Напівжирний </B> текст </P>
</BODY>
</HTML>
```

11.3. Доступ до об'єктів на Web-сторінці

Існує кілька способів доступу до об'єкта на Web-сторінці:

- за номером об'єкта в колекції document.all;
- за ідентифікатором об'єкта;
- щодо відзначення об'єкта мишкою.

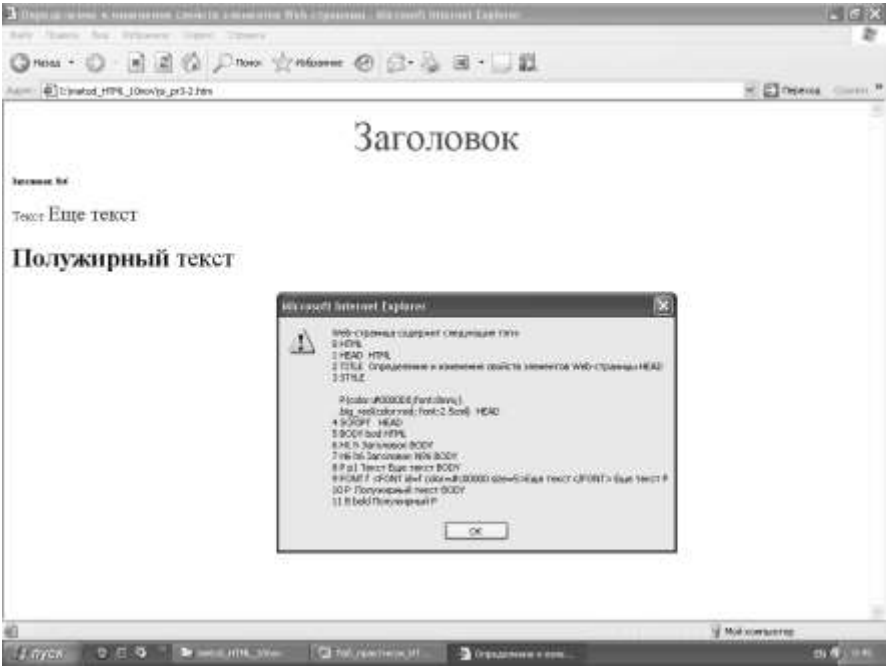


Рисунок 11.2 – Результат прикладу 11.2 на екрані

11.3.1. Доступ до об'єкта за його номером

Кожен тег на Web-сторінці і, відповідно, у колекції document.all, має свій номер. Теги нумеруються у тому порядку, в якому вони зустрічаються у HTML-документі. Наприклад, тег <HTML> має номер 0, тег <HEAD> − 1 і т. д. Для доступу до об'єкта за його номером використовується метод item() колекції document.all. Наприклад, доступ до і-го об'єкту здійснюється так: `document.all.item(i)`. Допускається і більш коротка форма цього вираження: `document.all(i)`.

Наприклад, використовуючи властивість outerHTML тега з номером 0, тобто <HTML>, можна вивести на екран весь зміст HTML-документа:

```
alert (document.all [0]. outerHTML);
```

Звернення до тегів за їх номерами для отримання їх властивостей показано також у ***прикладі 11.2***. Для цього використовується функція allTags(), яка викликається подією Load при завантаженні Web-сторінки.

```
      function allTags()
      {
      tags="Web-сторінка містить наступні теги:";
      for (i=0; i<document.all.length; i++)
      with (document.all.item(i))//Метод item()не обов'язковий
      {
      tags+="\n"+i+" "+tagName+" "+id;
      if (tagNam=="STYLE") tags+=" "+innerHTML;
      if (tagNam=="FONT") tags+=" "+outerHTML;
      if (tagName!="HTML"&&tagName!=BODY""&&tagName!="HEAD")
      tags+=" "+innerText;
      if (tagName!="HTML") tags+=" "+parentElement.tagName;
      }
alert(tags);
}
```

При виконані цієї функції за допомогою циклу for перебираються усі об'єкти колекції document.all, тобто усі теги Web-сторінки, і для кожного з них на екран за допомогою методу alert() виводяться такі загальні властивості об'єкта:

- номер тега – i (можна також використовувати властивість sourceIndex);
- найменування тега – tagName;
- ідентифікатор тега – id;
- текстовий вміст елемента Web-сторінки – innerText (для всіх тегів, крім <HTML> і <BODY> – щоб уникнути дублювання, оскільки ці елементи містять тексти всіх інших елементів);
- найменування тега-батька (для всіх тегів, крім тега <HTML>) – parentElement.tagName.

Зверніть увагу, що для властивості tagName найменування HTML-параметрів вказуються тільки великими буквами.

У функції allTags() застосований оператор with(). Він використовується для більш компактної записи виразів, які містять загальні

частини. У функції allTags() такої загальної частиною є вираз document.all.item(i), яке і вказується в якості аргументу оператора with(). Інакше довелося б замість, наприклад, tagName вказувати document.all.item(i). TagName.

11.3.2. Доступ до об'єкта за його ідентифікатором

У цьому випадку може застосовуватися одна з трьох форм використання ідентифікатора:

- document.all.item("ідентифікатор") або document.all("ідентифікатор");
- document.getElementById("ідентифікатор");
- ідентифікатор,

де ідентифікатор − це ідентифікатор тега, заданий за допомогою параметра ID, наприклад, txt в тегу `<P ID="txt">`.

Для тих тегів, для яких допускається використання параметра NAME, наприклад, таких як <IMG>, <A>, а також елементів тега <FORM>, як ідентифікатора замість параметра ID можна використовувати параметр NAME.

Третя форма доступу до тегу за його ідентифікатором є найбільш простою, її підтримує браузер Microsoft Internet Explorer, але до її недоліків слід віднести те, що не усі браузери підтримують цю форму доступу. Якщо необхідно добитися того, щоб Web-сторінка була правильно оброблена будь-яким браузером, краще скористатися другою, більш універсальною формою.

Необхідно звернути увагу на відмінність у двох варіантах позначення імені тега: в лапках − "ідентифікатор" і без лапок − ідентифікатор. Ім'я ідентифікатора тега у лапках має строковий тип та визначає саме ідентифікатор тега, тоді як ім'я ідентифікатора тега без лапок має тип об'єкта і задає вказівник на тег, як об'єкт.

Метод document.getElementById() повертає покажчик на тег, і присвоївши його значення змінної, наприклад, `h=document.getElementById()`,

можна далі працювати зі змінною h так само, як з ідентифікатором, зазначеним у третій формі.

Звернення до тегів за їх індифікаторами для зміни і відновлення властивостей об'єктів показано у ***прикладі 11.2.*** Для цього використовуються функції changeTag(obj) і restoreTag(obj).

Функція changeTag (obj)

```
function changeTag(obj)//Змінення властивостей параграфа
{
bod.background="back.gif";
obj.align="center";
obj.style.fontStyle="italic";
obj.className="big_red";
}>
```

викликається по натисненню лівої або правою клавіш мишки, тобто по події Mousedown, після наведення вказівника мишки на параграф з ідентифікатором "p1". Це відбувається тому, що саме у цей тег поміщений обробник події. З цієї ж причини параграф з ідентифікатором "p2" залишиться без зміни.

При виконанні функції в якості її параметра передається покажчик на об'єкт, заданий методом document.getElementById ('p1') або просто getElementById ('p1'):

```
onMousedown="changeTag(getElementById('p1'))"
```

При використанні браузера Microsoft Internet Explorer як параметр функції changeTag () можна вказати тільки ім'я ідентифікатора:

```
onMousedown="changeTag(p1)"
```

або змінну this, що вказує на поточний об'єкт:

```
onMousedown="changeTag(this)"
```

Функція changeTag(obj) виконує наступні дії:

- міняє фон Web-сторінки (дія, яка показує, що можна змінювати властивості не тільки того тега, який зазначений, але й усіх інших);
- виконує центрування параграфа шляхом зміни значення HTML-параметра ALIGN;
- за допомогою параметра STYLE змінює значення властивості CSS font-style;

- за допомогою тега <STYLE> змінює колір і розмір символів (значення властивостей CSS color і font-size).

Функція restoreTag(obj)

```
function restoreTag(obj)//Восстановление свойств
{
bod.background="";
obj.align="";
obj.style.fontStyle="";
obj.className="";
}
```

викликається по події Contextmenu, тобто по віджатію правої клавіші мишки при наведенні вказивніка миші на параграф з ідентифікатором "p1".

Виклик цієї функції відрізняється від описаного вище виклику функції changeTag(obj) тим, що для події Contextmenu вже існує заданий за умовчанням обробник події, який після дій, передбачених обробником, заданими у тегу, буде виконувати дії за умовчанням: виводити на екран контекстное меню. Щоб скасувати ці дії непобходімо в обробнику після виклику функції restoreTag (obj) вказати return false:

```
onContextmenu="restoreTag(getElementById('p1'));return
false"
```

Іншим варіантом скасування дії за умовчанням є включення в опис функції вираження:

```
event.returnValue=false;
```

(В цьому випадку в обробнику події вже не потрібно вказувати return false).

Функція restoreTag(obj) відновлює усі вихідні значення властивостей тега <P> шляхом задання усім цим властивостям порожніх значень ("").

11.3.3. Доступ до зазначеного об'єкту

Недоліком розглянутих вище звернень до об'єкта за його номером або за його ідентифікатором є те, що для цього необхідно знати його номер або його ідентифікатор. Цього недоліку позбавлений третій спосіб: доступ до об'єкта шляхом його відмітки мишкою.

Для того, щоб відзначити тег, властивості якого необхідно визначити або змінити, досить підвести до нього курсор миші і виконати одну з подій, яка здійснюються за допомогою мишки.

Доступ до зазначених елементів і його властивостям здійснюється за допомогою вираження:

```
window.event.srcElement.властивість,
```

де window.event.srcElement - зазначений об'єкт; властивість − одна з його властивостей (див. підрозділ 11.1.3. "Об'єкт event").

Звернення до зазначених елементів Web-сторінки для визначення їх властивостей показано у ***прикладі 11.2***. Для цього використовується зовнішня функція showTagInfo(),

```
function showTagInfo()
{
with (window.event.srcElement)
{
window.status=
"подія: "+event.type+
"номер тега: "+sourceIndex+
"найменування: "+tagName+
"колір: "+ currentStyle.color +
"розмір: "+ currentStyle.fontSize+
"жирність: "+currentStyle.fontWeight;
}
}
```

яка викликається при клацанні мишки на будь-якому об'єкті Web-сторінки (по події Click), виводить ім'я цієї події і визначає значення таких властивостей зазначеного об'єкту:

- номер (загальна властивість);
- ім'я (загальна властивість);
- колір (задається тегом <STYLE> і параметром COLOR тегу <FONT>);

- розмір (задається тегом <STYLE>, параметром STYLE і параметром SIZE тегу <FONT>);
- жирність (задається тегом <B>).

При візначенні властивостей кольору, розміру, жірності та стилю було використано властівість currentStyle, яка дозволяє визначати властивості об'єктів, задані HTML-параметрами, тегами STYLE і параметрами STYLE .

Для виведення знайдених значень властивостей об'єкта функція showTagInfo() використовує властивість status об'єкта window, яка дозволяє вивести дані у полі статусу вікна браузера. Наприклад, при кліку мишкою по слову Напівжирний в полі статус виводяться наступні дані: Подія: `click` номер: `10` ім'я: `B` колір: `#0000d0` розмір: `8mm` жирність: `700`.

Якщо зазначений об'єкт необхідно використовувати не відразу, то можна при позначці запам'ятати його номер за допомогою властивості sourceIndex:

```
onClick="num=window.event.srcElement.sourceIndex".
```

Надалі доступ до зазначеного об'єкту з номером num можна здійснити наступним чином:

```
document.all(num).властивість.
```

11.4. Колекції об'єктів

Окрім розглянутої вище колекції document.all, яка містить всі об'єкти Web-сторінки, JavaScript підтримує також декілька колекцій об'єктів, які є складовими частинами загальної колекції, кожна з яких містить усі теги HTML-документа одного найменування:

- forms − форми;
- frames − фрейми;
- images − рисунки;
- links − гіперпосилання;
- scripts − скрипти.

Колекції зручно використовувати, коли необхідно звернутися для зміни властивостей не до одного тегу на Web-сторінці, а до усіх тегів або групі тегів одного найменування. Доступ до елементів цих колекцій здійснюється так само, як до колекції document.all: або за номерами елементів, або ідентифікаторами елементів. Наприклад до першого рисунку на Web-сторінці можна звернуться так: `document.images [0].`

Крім існуючих колекцій можна створювати колекції користувача. Для цього є два однакових метода за виконуваними функціями, кожен з яких повертає колекцію тегів даного найменування:

- `document.getElementByName("найменування тега");`
- `document.all.tags("найменування тега").`

У ***прикладі 11.2*** показано створення колекції параграфів (тегів <P>) Web-сторінки і зміна їх розмірів символів.

Для цього використовується зовнішня функція changeCollection(),

```
function changeCollection()
{
tagsP=document.getElementsByTagName("P")
for(i=0;i<tagsP.length;i++)
{
size=parseInt(tagsP[i].currentStyle.fontSize);
tagsP[i].runtimeStyle.fontSize=size*1.25+"mm";
}
}
```

що викликається за подвійним клацання мишки (за подією Dblclick) та виконує наступні дії:

- за допомогою методу document.getElementsByTagName() створює колекцію з ім'ям tagsP усіх параграфів Web-сторінки (в прикладі їх два);
- для кожного елемента колекції tagsP за допомогою властивості currentStyle визначає поточний розмір символів (властивість font-size) і за допомогою властивості runtimeStyle збільшує його в 1,25 рази.

Індивідуальні завдання

На мові JavaScript розробити програму, яка виконує згідно з вказаними у таблиці 11.1 індивідуальними параметрами, наступні дії:

- за подією 1 за допомогою зазначеного способу доступу до тегу змінити його властивості (властивості повинні бути зазначені:
  - у вигляді HTML-параметра;
  - за допомогою параметра STYLE;
  - з допомогою тега <STYLE>).
- за подією 2 відновити початкові властивості тега. При цьому використовувати альтернативний спосіб доступу до тегу (якщо при зміні властивостей застосовувався доступ по відмітке, то тепер − за ідентифікатором, і навпаки).
- за подією 3 за допомогою колекції змінити одну властивість усіх наявних на Web-сторінці тегів (не менше трьох) зазначеного найменування.

Таблиця 11.1 – Варіанти індивідуальних завдань

№	Спосіб доступу	Подія 1	Подія 2	Тег	Подія 3	Теги
1	2	3	4	5	6	7
1	по відмитке	Mouseover	Mouseout	H1	Click	IMG
2	по відмитке	Mousemove	Click	SPAN	Contextmenu	H1
3	по відмитке	Mousedown	Mouseup	OL	Contextmenu	SPAN
4	по відмитке	Click	Mousemove	UL	Contextmenu	TABLE
5	по відмитке	Dblclick	Mouseout	TABLE	Contextmenu	OL
6	по відмитке	Mousemove	Mouseout	H2	Dblclick	UL
7	по відмитке	Mouseover	Contextmenu	TR	Dblclick	H2
8	по відмитке	Mousemove	Dblclick	TH	Contextmenu	DIV
9	по відмитке	Mousedown	Mouseout	TD	Contextmenu	IMG
10	по відмитке	Click	Mouseout	H3	Contextmenu	SPAN
11	по ідентифікатору	Dblclick	Mouseout	IMG	Contextmenu	TABLE
12	по ідентифікатору	Contextmenu	Mouseout	A	Click	OL

Продовженя табл. 11.1

1	2	3	4	5	6	7
13	по ідентифікатору	Mouseover	Click	FRAME	Dblclick	UL
14	по ідентифікатору	Mousemove	Dblclick	IFRAME	Click	H3
15	по ідентифікатору	Mousedown	Mouseup	INPUT	Mousemove	DIV
16	по ідентифікатору	Click	Mousedown	TEXTARREA	Dblclick	IMG
17	по ідентифікатору	Dblclick	Mouseover	SELECT	Contextmenu	A
18	по ідентифікатору	Contextmenu	Mousemove	H4	Click	SPAN
19	по ідентифікатору	Mouseover	Dblclick	SPAN	Contextmenu	TABLE
20	по ідентифікатору	Mousemove	Dblclick	H5	Contextmenu	H4

ЛАБОРАТОРНА РОБОТА 12
РУХ ЕЛЕМЕНТІВ НА WEB-СТОРІНЦІ

Мета: Вивчення засобів і можливостей мови JavaScript для організації руху елементів на Web-сторінці

12.1. Організація руху об'єктів на Web-сторінці

Рух елемента на Web-сторінці здійснюється шляхом зміни значень властивостей, що визначають його координати. Властивості CSS, за допомогою яких можна задавати режим позиціонування елемента і його координати на екрані, розглянуті в підрозділі 5.1.3 лабораторної роботи 5 "Розміщення зображень на Web-сторінці".Нижче наведено приклад рисунка, розташованого у 200 пікселях від лівої межі екрана і у 100 пікселях від верхньої межі екрана:

```
<IMG NAME="imgs" SRC="fish.gif"
STYLE="position: absolute; left: 200; top: 100">
```

Якщо спробувати зрушити цей рисунок на екрані на 50 пікселів вправо за допомогою виразу

```
imgs.style.left +=50,
```

то це призведе до виникнення помилки "Неприпустимий аргумент" через те, що властивість left повертає значення не у вигляді числа 200, а у вигляді рядка "200px".

Тому треба або скористатися функцією parseInt():

```
imgs.style.left=parseInt (imgs.style.left) +50,
```

або замість властивостей left, top, right і bottom застосовувати властивості:

- pixelLeft – задає або повертає чисельне значення горизонтальної позиції лівої межі елемента в пікселях;
- pixelTop – задає або повертає чисельне значення вертикальної позиції верхньої межі елемента в пікселях;
- pixelRight – задає або повертає чисельне значення горизонтальної позиції правої межі елемента в пікселях;

- pixelBottom – задає або повертає чисельне значення вертикальної позиції нижньої межі елемента в пікселях.

У нашому прикладі замість властивості left необхідно використовувати властивість pixelLeft:

```
imgs.style.pixelLeft +=50;
```

Якщо ж координати елемента задані в інших одиницях виміру, необхідно використовувати властивості:

- posLeft – задає або повертає чисельне значення горизонтальної позиції лівої межі елемента в одиницях виміру, заданих властивістю left;
- posTop – задає або повертає чисельне значення вертикальній позиції верхньої межі елемента в одиницях виміру, заданих властивістю top;
- posRight – задає або повертає чисельне значення горизонтальної позиції правої межі елемента в одиницях виміру, заданих властивістю right;
- posBottom – задає або повертає чисельне значення вертикальній позиції нижньої межі елемента в одиницях виміру, заданих властивістю bottom.

Зміна координат елемента може бути реалізована:

- шляхом програмно організованого циклічного процесу (див. ***приклад 12.1***);
- щодо подій мишки (див. ***приклад 12.2***);
- шляхом "прив'язки" елемента до курсора мишки, тобто шляхом задання координат елемента, що дорівнюють координатам курсору мишки (див. ***приклад 12.3***);
- щодо подій клавіатури (див. опис лаб. роботи 13 "Робота з клавіатурою").

12.1.1. Зміна координат елемента за допомогою циклічного процесу

Для організації циклічного процесу можуть бути використані такі методи об'єкта window:

- setTimeout(функція чи вираз, інтервал [, список аргументів функції, розділених комами]): обчислює значення виразу або викликає функцію після закінчення заданого інтервалу (у мілісекундах), якщо до цього не був викликаний метод clearTimeout (), може передавати у функцію задані у

списку аргументи, повертає вказівник на об'єкт таймера, який може бути використаний в методі clearTimeout() для зупинки і знищення таймера;

- clearTimeout(таймер): зупиняє таймер, установлений методом setTimeout();
- setInterval(функція чи вираз, інтервал [, список аргументів функції, розділених комами]): обчислює значення виразу або викликає функцію кожного разу після закінчення даного інтервалу (в мілісекундах), якщо до цього не був викликаний метод clearTimeout(), може передавати у функцію задані в списку аргументи, повертає вказівник на об'єкт таймера, який може бути використаний в методі clearTimeout() для зупинки і знищення таймера;
- clearInterval(таймер): зупиняє таймер, установлений методом setInterval().

Використання методу setTimeout() для організації руху елемента показано у ***прикладі 12.1***, де реалізовано рух елемента Web-сторінки − букви Z, заданої за допомогою тега <P>, по траєкторії у вигляді синусоїди ():

Приклад 12.1

```
<HTML>
<HEAD>
<TITLE> Изменение координат элемента с помощью
циклического процесса </TITLE>
<SCRIPT>
T=0; X=0; Y=200;
function f(x) {return60*Math.sin(x*Math.PI/180);}
function moveTxt()
{
if ((z.style.pixelLeft<document.body.clientWidth-25)&&
z.style.pixelTop>25))
{
z.style.left=X+T;
z.style.top=Yf(T);
T+=25;//Швидкість руху
setTimeout ("moveTxt()",125);
}
}
</SCRIPT>
```

```
      </HEAD>
      <BODY>
      <P  ID="z"  STYLE="color:blue;font  8mm;position:absolute;
left:0;top:200"
      onClick="moveTxt()">Z</P>
      </BODY>
      </HTML>
```

Рисунок 12.1 – Результат прикладу 12.1 на екрані

У ***прикладі 12.1*** за допомогою функції f (x) реалізовано рух об'єкта справа наліво і зверху вниз по синусоїді y = 60sin (x). Оскільки траєкторія вказана у вигляді тригометріческой функції, для її завдання був використаний об'єкт Math, властивості і методи якого описані в розділі 12.2. При цьому було виконано перетворення аргументу функції sin (x) з градусів в радіани. Напрямок та швидкість руху об'єкта залежить від величини і знака збільшення його координат dx і dy. На швидкість руху також впливає проміжок часу, через який ці збільшення відбуваються (2-й параметр методу setTimeout ()).

Вибір амплітуди і швидкості руху в прикладі 1 здійснено з міркувань наочності.

По клацанню мишки по букві Z викликається функція moveTxt(), яка починає виконувати рух цього елемента з точки екрана з координатами: left=0; top=200 шляхом задання збільшень його координатами left і top:

```
z.style.left=X+T;
z.style.top=Yf(T);
```

і завершує цей рух поблизу межі екрана (межа не досягається, щоб літера Z залишалася в області повної видимості).

Ліва та нижня межі екрана визначаються за допомогою властивостей тега <BODY> clientWidth і clientHeight. Ці властивості можуть бути використані для визначення розмірів будь-яких елементів Web-сторінки, які мають ширину і висоту.

При використанні альтернативних методів setInterval() і clearInterval() опис функції moveTxt() необхідно дещо змінити:

```
function moveTxt()
{
if ((z.style.pixelLeft<document.body.clientWidth-25)&&
(z.style.pixelTop>25))
{
z.style.left=X+T;
z.style.top=Yf(T);
T+=25; // Швидкість руху
)
else clearInterval(timer);
}
```

Крім того, виклик функції moveTxt() у цьому випадку слід виконувати так:

```
onClick="timer=setInterval('moveTxt()',125);"
```

12.1.2. Зміна координат елемента щодо подій мишки

Рух елемента Web-сторінки по екрану крім розглянутого вище способу зміни його координат за допомогою програмно організованого циклічного процесу, може здійснити сам користувач Web-сайту за допомогою мишки.

У ***прикладі 12.2*** описана функція runRef(), яка кожного разу при досягненні вказівником мишки посилання "Посилання" або при руху вказивніка мишки по посиланню виконує приросту координат цього елемента, реалізуючи ефект "тікає" від користувача посилання. Усі його спроби натиснути на посилання, щоб завантажити нову сторінку (js_1.htm), виявляються безрезультатними.

За допомогою змінних dx і dy, які змінюють свій знак на протилежний при досягненні відповідно вертикальних або горизонтальних меж екрана, реалізовано відображення посилання при руху по екрану від його меж.

Виклик функції runRef() як за подією Mousemove, так і за подією Mouseover забезпечує більш надійний захист посилання від клацання мишки.

Приклад 12.2

```
<HTML>
<HEAD>
<TITLE>Организация движения элементов Web-страницы с
помощью мышки </TITLE>
<SCRIPT>
dx=10; dy=10;
function runRef()
{
if ((parseInt(m.style.left)<25) ||
(parseInt(m.style.left)>document.body.clientWidth-25))
dx=-dx;
if ((m.style.posTop<25) ||
(m.style.posTop>document.body.clientHeight-25)) dy=-dy;
m.style.pixelLeft+=dx;
m.style.pixelTop+=dy;
}
</SCRIPT>
</HEAD>
<BODY">
<A NAME="m" HREF="js_1.htm"
```

```
STYLE="position:absolute;left:100;top:100;color:red"
onMousemove="runRef()"
onMouseover="runRef()"> Ссылка </A>
</BODY>
</HTML>
```

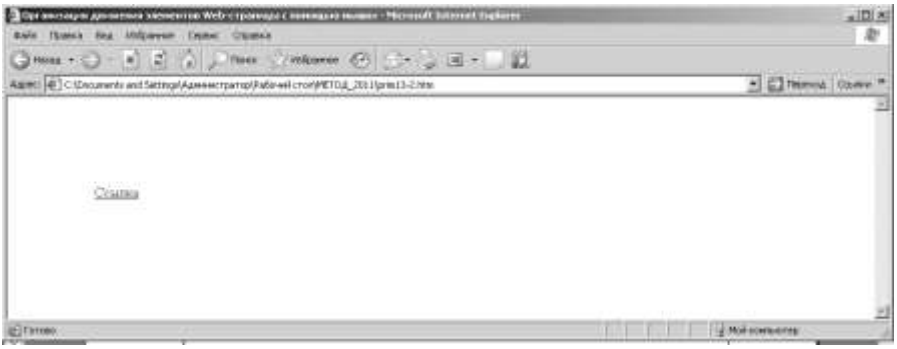


Рисунок 12.2 – Результат прикладу 12.2 на екрані

12.1.3. Визначення координат мишки

Доступ до координат мишки здійснюється дещо по-різному в підході, запропонованому фірмою Microsoft, і в рекомендаціях організації W3C. Браузер Internet Explorer для цієї мети використовує такі властивості об'єкта event:

- clientX − повертає горизонтальну координату курсора мишки відносно клієнткої частині вікна, тобто без урахування рамок, заголовка, рядка меню, панелі инстументов і рядка стану (це властивість в W3C має таке ж значення);
- clientY − повертає вертикальну координату курсора мишки відносно клієнткої частині вікна, тобто без урахування рамок, заголовка, рядка меню, панелі инстументов і рядка стану (це властивість в W3C має таке ж значення);
- offsetX − повертає горизонтальну координату курсора мишки відносно позиціонованого (absolute або relative) елемента сторінки, що викликав цю подію; якщо курсор мишки знаходиться поза елементом − то щодо документа (властивість з аналогічним значенням в W3C називається layerX);
- offsetY − повертає вертикальну координату курсора мишки щодо позиціонованого (absolute або relative) елемента сторінки, що викликав цю подію; якщо курсор мишки знаходиться поза елементом − то щодо документа (властивість з аналогічним значенням в W3C називається layerY);

- screenX − повертає горизонтальну координату курсора мишки відносно екрану (це властивість в W3C має таке ж значення);
- screenY − повертає вертикальну координату курсора мишки відносно екрану (це властивість в W3C має таке ж значення);
- x − повертає горизонтальну координату курсора мишки відносно батьківського елементу (за замовчуванням <BODY>);
- y − повертає вертикальну координату курсора мишки відносно батьківського елементу (за замовчуванням <BODY>).

Доступ до координат мишки можна здійснити при використані рекомендацій W3C, використовуючи такі властивості дії:

- clientX – повертає горизонтальну координату курсора мишки щодо клієнтської частини вікна (без урахування рамок, заголовка, рядка меню, панелі інстументов і рядка стану);
- clientY – повертає вертикальну координату курсора мишки щодо клієнтської частини вікна (без урахування рамок, заголовка, рядка меню, панелі інстументов і рядка стану);
- offsetX – повертає горизонтальну координату курсора мишки щодо елемента сторінки, який з'єднав цю подію;
- offsetY – повертає вертикальну координату курсора мишки щодо елемента сторінки, який з'єднав цю подію;
- screenX – повертає горизонтальну координату курсора мишки щодо екрана;
- screenY – повертає вертикальну координату курсора мишки щодо екрана;
- pageX – повертає горизонтальну координату курсора мишки щодо документу (при горизонтальному прокручуванні), еквівалентно pageX = window.pageXOffset+e.clientX;
- pageY – повертає вертикальну координату курсора мишки щодо документу (при вертикальному прокручуванні), еквівалентно pageY = window.pageYOffset+e.clientY.

Оскільки Internet Explorer не підтримує властивостей pageX і pageY, то для забезпечення сумісності браузерів при визначенні координат мишки щодо документа можна використовувати наступну функцію:

```
      function posMouse(e) { //Определение координат мышки
относительно документа
      var posX, posY;
      if( e ) { // для браузеров с поддержкой W3C
      if (e.pageX || e.pageY) {
      posX=e.pageX;
      posY=e.pageY;
      }
      else {
      posX=e.clientX; // для браузера Opera
      posY=e.clientY;
      }
      }
      else { // для браузера Internet Explorer
      e = window.event;
      posX = e.clientX;
      posY = e.clientY;
      var html = document.documentElement;
      var body = document.body;
      if( (html && html.scrollLeft) ||(html &&
html.scrollTop)) {
      posX += html.scrollLeft;
      posY += html.scrollTop;
      }
      if( (body && body.scrollLeft) || (body &&
body.scrollTop)) {
      posX += body.scrollLeft;
      posY += body.scrollTop;
      }
      }
      window.status="posX= "+posX+"posY= "+posY;
      }
```

12.1.4. Перетягування елементів

Перетягування елементів Web-сторінки за допомогою мишки реалізується шляхом присвоювання поточним координатами поточних координат мишки переміщуваного елемента.

У ***прикладі 12.3*** створюється Web-сторінка, яка містить три елементи − рисунок і два слова "ТЕКСТ" однакового розміру, але різних кольорів.

Приклад 12.3

```
<HTML>
<HEAD>
<TITLE>Перетаскивание элементов Web-страницы</TITLE>
<SCRIPT>
var n=0,l=0;
function drag()
{
with (document.all(n).style)
{
/* window.status= "n="+n+" z-index="+l+"
left="+pixelLeft+" top="+pixelTop+
" clientX="+event.clientX+" clientY="+event.clientY; */
position="absolute";
left=event.clientX;
top=event.clientY;
zIndex=l;
}
}
</SCRIPT>
</HEAD>
<BODY onClick="n=event.srcElement.sourceIndex"
onContextmenu="if(l==2) l=0; else l++;return false"
onMousemove="drag()">
<IMG SRC="fish.gif" >
<P STYLE="color:#0FF;font:8mm">TEXT1
<P STYLE="color:#F0F;font:8mm">TEXT2
</BODY>
</HTML>
```

Рисунок 12.3 – Результат прикладу 12.3 на екрані

При клацанні мишки по будь-якому з цих об'єктів (по події Click) для зазначеного об'єкта встановлюється режим перетягування: в змінну n заноситься порядковий номер цього елемента Web-сторінки, наприклад, для рисунка: n = 5, для першого тексту: n = 6, для другого: n = 7 (при клацанні по вільному простору вікна браузера n набуває значення 4 - номер тега BODY).

Режим перетягування зазначеного об'єкта реалізується шляхом виклику при руху мишки (по події Mousemove) функції drag(), яка виконує такі дії:

- встановлює для об'єкта з номером n абсолютне позиціювання;
- присвоює горизонтальній і вертикальній координатам цього об'єкта відповідно горизонтальну і вертикальну координати вказивніка мишки, що дозволяє переміщатися за мишкою, яка рухається;
- встановлює для об'єкта поточне значення властивості z-index, задане у змінній l.

При клацанні правою кнопкою мишки (по події Contextmenu) циклічно змінюється значення змінної l (0,1,2,0 ,...), яка задає поточне значення властивості z-index, що дозволяє будь-якому об'єкту при заданні відповідного значення або перекривати інші об'кти Web -сторінки, або розташовуватися під ними.

Для скасування режиму перетягування зазначеного об'єкта необхідно повторно клацнуть мишкою. При цьому в змінну n заноситься номер вже нового об'єкта – того, який у момент клацання знаходився під перетягуваним об'єктом. Це пов'язано з тим, що при руху координати вказивніка миші на

піксель випереджають координати самого об'єкта. У цьому випадку для нового об'єкта встановлюється режим перетягування (якщо тільки цим об'єктом не є тег BODY).

Додамо, що закоментовані рядки функції drag() були використані при налагодженні програми для виведення у полі статусу вікна браузера значень таких контрольних даних: номер, z-index і координати перетягуванного об'єкта, а також координати мишки.

12.2. Властивості та методи об'єкта Math

Об'єкт Math дозволяє програмістам використовувати математичні константи та функції при написанні програм на JavaScript. Має такі характеристики:

- E – повертає константу Ейлера (e);
- LN10 – повертає значення ln10;
- LN2 – повертає значення ln2;
- LOG10E – повертає значення lg e;
- LOG2E – повертає значення log2e;
- PI – повертає значення 3,14159 ...

і методи:

- abs (число) – повертає абсолютне значення аргументу;
- acos (число) – повертає арккосинусу аргументу у радіанах;
- asin (число) – повертає синус аргументу у радіанах;
- atan (число) – повертає арктангенс аргументу у радіанах;
- atan2 (x, y) – повертає кут у радіанах між горизонтальною віссю і прямою, проведеною через початок координат і точку з координатами x, y;
- ceil (число) – повертає найближчим ціле число, більше або рівне аргументу;
- cos (число) – повертає косинус аргументу у радіанах;
- exp (число) – повертає значення $e^{число}$;
- floor (число) – повертає найближчим ціле число, менше аргументу або рівне йому;
- log (число) – повертає натуральний логарифм аргументу;

- max (список аргументів, розділених комами) – повертає максимальний з аргументів; якщо не заданий ні один аргумент, повертає значення NEGATIVE INFINITY ("мінус нескінченність"); якщо один з аргументів дорівнює NaN (Not a Number – "не число"), повертається NaN ;
- min (список аргументів, розділений комами) – повертає мінімальний з аргументів; якщо не заданий ні один аргумент повертає значення POSITIVE INFINITY ("плюс нескінченність"); якщо один з аргументів дорівнює NaN, повертається NaN;
- pow (підстава, порядок) – повертає значення підстава^{порядок};
- random ()– повертає псевдовипадкове число від 0 включно до 1 виключно;
- round (число) – повертає значення аргументу, округлене до найближчого цілого;
- sin (число) – повертає синус аргументу в радіанах;
- sqrt (число) – повертає квадратний корінь від аргументу;
- tan (число) – повертає тангенс аргументу в радіанах.

Індивідуальні завдання

На мові JavaScript розробити програмні засоби, які виконують такі дії:

- за подією 1 вказати зазначений у таблиці об'єкт, змінивши одну або декілька його властивостей;
- за подією 2 здійснити рух цього об'єкта з заданої точки (x, y) екрана по траєкторії, заданої функцією f(x) і напрямком ("куди") згідно із зазначеним режимом:
  - 1 – до досягнення меж вікна браузера;
  - 2 – до досягнення меж вікна браузера після одноразового відбиття;
  - 2 – безперервний рух з багаторазовими відбитками від меж вікна браузера, треба виконувати з відбиттям від меж вікна браузера.
- амплітуду та швидкість руху вибирати з міркування наочності.

Таблиця 12.1 – Варіанти індівідуальних завдань

№	Об'єкт	Параметри руху				Події	
		(x,y)	f(x)	куди	ре-жим	1	2
1	Слово	X=0;Y=450	2*x+5	праворуч/ вгору	3	Click	Click
2	Рисунок	X=300;Y=400	25*cos(x)	вгору	2	Click	Contextmenu
3	посилання	X=450;Y=220	40*sin(3*x)	вліво	2	Click	Mouseover
4	Таблиця	X=600;Y=400	x*x/500	вліво/ вгору	3	Click>	Mousemove
5	Буква	X=200;Y=0	exp(x/100)	вниз	2	Dblclick	Dblclick
6	рисунок	X=10;Y=300	30*log(x+10)	праворуч	2	Dblclick	Contextmenu
7	посилання	X=600;Y=200	10*tan(x)	вліво	2	Dblclick	Mouseover
8	Слово	X=50;Y=50	0.001*x*x+x	праворуч/ вниз	3	Dblclick	Mousemove
9	Кнопка	X=500;Y=5	60*sin(x*x)	вліво/вниз	1	Contextmenu	Contextmenu
10	Буква	X=250;Y=450	25*cos(2x+1)	вгору	2	Contextmenu	Click
11	посилання	X=350;Y=0	0.5*exp(x/250)	вниз	2	Contextmenu	Dblclick
12	посилання	X=75;Y=175	50*log(x+5)-50	праворуч	3	Contextmenu	Mouseover
13	Форма	X=20;Y=420	0.2*x+45	праворуч/ вгору	3	Contextmenu	Mousemove
14	список OL	X=550;Y=20	-2.5*x+15	вліво/ вниз	1	Mouseover	Click
15	Число	X=650;Y=510	1000/(x+25)	вліво/ вгору	1	Mouseover	Dblclick
16	IFRAME	X=20;Y=80	x*x*x/ (x*x+125))	праворуч/ вниз	3	Mouseover	Contextmenu
17	список UL	X=20;Y=330	x*0.75-15	праворуч	2	Mouseover	Mouseout
18	Radio	X=620;Y=280	x*1.05-x	вліво	2	Mousemove	Click
19	checkbox	X=20;Y=440	x*0.5-50	праворуч/ вгору	3	Mousemove	Dblclick
20	SELECT	X=580;Y=590	(x*x+x+10)0.5	вліво/ вгору	2	Mousemove	Contextmenu

ЛАБОРАТОРНА РОБОТА 13
ПРИХОВУВАННЯ ЕЛЕМЕНТІВ WEB-СТОРІНКИ. РОБОТА З КЛАВІАТУРОЮ

Мета: Вивчення засобів і можливостей мови JavaScript для приховування елементів Web-сторінки і обробки подій, пов'язаних з роботою користувача на клавіатурі.

13.1. Приховування елементів Web-сторінки

Щоб приховати елементи Web-сторінки, тобто для того, щоб зробити ці елементи невидимими, а також для того, щоб відновити їх видимість на Web-сторінці, використовується дві властивості каскадних листків стилів: властивість CSS display і властивість CSS visibility.

13.1.1. Властивість CSS display

Властивість display визначає, як елемент повинен бути показаний на Web-сторінці, і мати наступні значення:

- block − елемент показується як блоковий, застосування цього значення для вбудованих елементів, наприклад тега <SPAN>, змушує його вести подібно блокам − додається новий рядок на початку і в кінці змісту тега;
- inline − елемент відображається як вбудований, використання блокових тегів, таких як <DIV> і <P>, автоматично показує зміст цих тегів з нового рядка. Аргумент inline скасовує цю особливість, тому зміст блокових елементів починається з того місця, де закінчився попередній елемент;
- none − тимчасово видаляє елемент з документа, займане ним місце не резервується і Web-сторінка формується так, наче елемента і не було. Змінити значення параметра і зробити його знову видимим можна за допомогою скриптів, звертаючись до властивостей через об'єктну модель. У цьому випадку відбувається переформатування даних на сторінці з урахуванням знову доданого елемента;
- пусте значення аргументу (прийняте браузером за замовчуванням) означає, що елемент буде видно на Web-сторінці.

Використання властивості CSS display показано у ***прикладі 13.1***, де, завдяки цій властивості, у текстовому документі, який містить кілька розділів, зміст кожної глави по клацанню миші урізається на екрані до назви глави і відновлюється при повторному клацанні миші.

Приклад 13.1

```
<HTML>
<HEAD>
<TITLE> Використання властивості display </TITLE>
<SCRIPT>
function showTxt()
{
with(window.event.srcElement)
if (tagName=="H1")
with(document.all.item(id+"p").style)
display=display? "" : "none";
}
</SCRIPT>
</HEAD>
<BODY OnClick="showTxt()">
<H1 ID="hd1" ALIGN=center"> Глава 1 </H1>
<P ID="hd1p" STYLE="display:none"> Текст першого розділу
Текст першого розділу Текст першого розділу Текст першого
розділу Текст першого розділу Текст першого розділу Текст
першого розділу Текст першого розділу Текст першого розділу
Текст першого розділу Текст першого розділу. . .
<H1 ID="hd2" ALIGN=center"> Глава 2 </H1>
<P ID="hd2p" STYLE="display:none"> Текст другого розділу
Текст другого розділу Текст другого розділу Текст другого
розділу Текст другого розділу Текст другого розділу Текст
другого розділу Текст другого розділу Текст другого розділу
Текст другого розділу Текст другого розділу. . .
<H1 ID="hd3" ALIGN=center"> Глава 3 </H1>
<P ID="hd3p" STYLE="display:none"> Текст третьої глави
Текст третьої глави Текст третьої глави Текст третьої глави
Текст третьої глави Текст третьої глави Текст третьої глави
Текст третьої глави Текст третьої глави Текст третьої глави
Текст третьої глави. . .
</BODY>
</HTML>
```

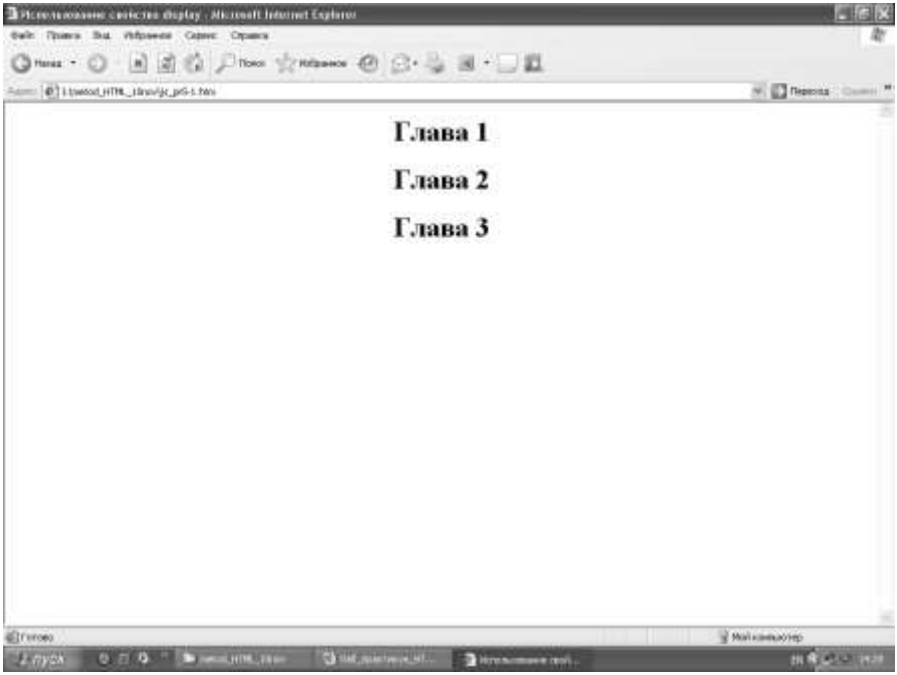


Рисунок 13.1 – Результата приклада 13.1 на екрані

За подією onClick викликається функція showTxt(), яка перевіряє найменування зазначеного тега. Якщо це тег <H1>, тобто клацання мишки було проведене за назвою глави, то значення властивості display для тега <P>, який має ідентифікатор, рівний ідентифікатору зазначеного тега плюс "p", зміниться на протилежне: якщо текст глави було не видно, він стане видимим і навпаки.

Використання оператора with дозволило двічі замінити громіздке вираження document.all.item (window.event.srcElement + "p"). Style.display словом display.

Єдиним обмеженням, яке накладається на цей HTML-документ, є те, що ідентифікатор тега <P> кожної глави повинен бути рівний ідентифікатору відповідного тега <H1> з додаванням буви "p".

13.1.2. Властивість CSS visibility

Властивість visibility призначена для відображення або приховування елементів Web-сторінки, включаючи рамку навколо нього і фон. При приховання елемента, хоча він і стає невидимим, місце, яке займає елемент, залишається за ним. Якщо передбачається вивід різних елементів у одне і те же місце екрану слід використовувати абсолютне позиціонування або скористатися властивістю display. Властивість visibility може приймати такі значення:

- visible - відображає елемент як видимий;
- hidden - елемент стає невидимим або, точніше, повністю прозорим, оскільки він продовжує брати участь у форматуванні сторінки;
- collapse - якщо це значення застосовується не до рядків або колонок таблиці, то результат його використання буде таким же, як hidden. У разі використання collapse для змісту осередків таблиць, то вони реагують, ніби до них було застосована властивість display зі значенням none, при цьому задані рядки і колонки прибираються, а таблиця перебудовується заново (цей аргумент не підтримується браузером MIE).

Використання властивості CSS visibility показано у ***прикладі 13.2*** , де за допомогою тегів <DIV> створюється двошарова структура, перший шар якої складають два елементи форми для введення тексту (теги <INPUT>), а другий складається з рисунка (тег <IMG>).

Приклад 13.2

```
<HTML>
<HEAD>
<TITLE> Використання властивості visibility </TITLE>
</HEAD>
<BODY TEXT=red>
<INPUT TYPE = button VALUE = "Шар 1" onClick =
    "lay1.style.visibility = 'visible';
    lay2.style.visibility = 'hidden'; ">
<INPUT TYPE = button VALUE = "Шар 2" onClick =
    "lay1.style.visibility = 'hidden';
    lay2.style.visibility = 'visible'; ">
<INPUT TYPE = button VALUE = "Шари 1 і 2" onClick =
    "lay1.style.visibility = 'visible';
    lay2.style.visibility = 'visible'; ">
```

```
      <DIV ID="lay1" STYLE="position:absolute;
left:200;top:100;width:200">
      <P> Ім'я: <INPUT TYPE=text NAME="user_name">
      <P> Тел.: <INPUT TYPE=text NAME="tel"> </DIV>
      <DIV ID="lay2" STYLE="position:absolute;
left:200;top:100; z-index:-1; visibility:hidden">
      <IMG SRC="fish.gif" WIDTH=200> </DIV>
      </BODY>
      </HTML>
```

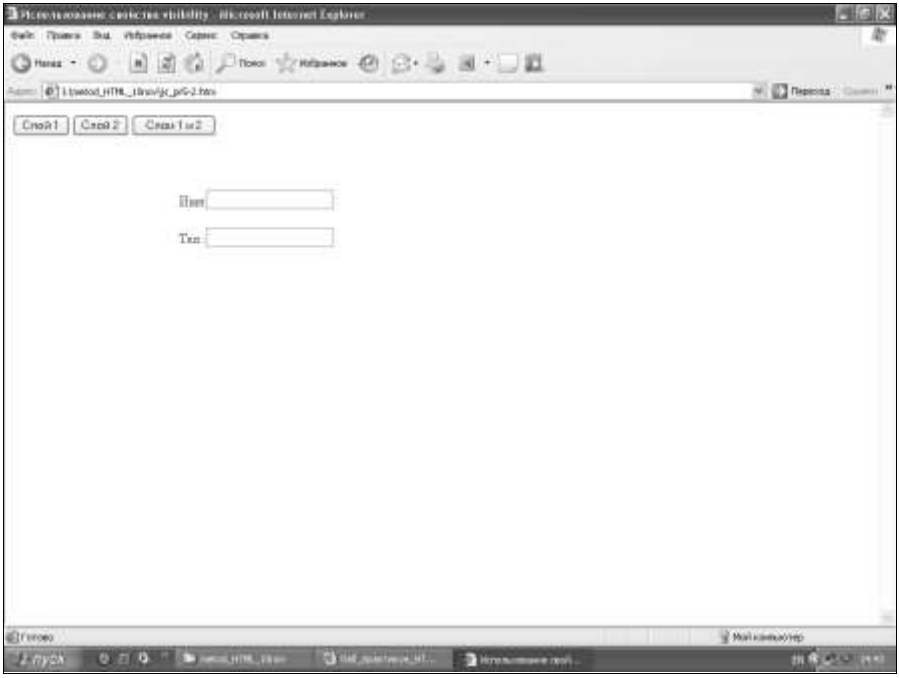


Рисунок 13.2 – Результата приклада 13.2 на екрані

Після завантаження Web-сторінки на екрані буде видно тільки 1-й шар, оскільки він за умовчанням має значення visibile, а для 2-го шару задано значення hidden.

При натисканні однієї з кнопок "Шар 1", "Шар 2" або "Шари 1 і 2" на екрані будуть відображатися відповідно перший шар, другий шар або два

шари одночасно, причому у останньому випадку рисунок виявиться знизу, так як цей шар має значення z-index: -1.

13.2. Робота з клавіатурою

При інтерактивноої взаємодії користувача з Web-сторінкою з використанням клавіатури виникають наступні події:

- keyDown − при натисканні клавіші;
- keyPress − при натисканні клавіші;
- keyUp − при відпусканні клавіші.

Події keyDown і keyPress розрізняються значеннями властивостей об'єкта event.

У лабораторній роботі 11 "Обробка подій і зміна властивостей об'єктів на Web-сторінці" були розглянуті властивості об'кта event, які пов'язані з роботою з мишкою. При виникненні подій, пов'язаних з роботою користувача з клавіатурою, об'єкт event формує та передає наступні властивості:

- keyCode − повертає значення Unicode клавіші (для цифр і латинських букв збігається з ASCII-кодом клавіші), яке має особливості, які залежать від типу події:

  − для подій keyDown і keyUp − повертає Unicode всіх клавіш, включаючи керуючі, для букв не залежить від стану регістра клавіатури і встановленого мови − завжди Unicode великих латинських букв;

  − для події keyPress − не реагує на натискання клавіш, що управляють, повертає Unicode літер з урахуванням регістра клавіатури (у верхньому регістрі, рядкові) і встановленої мови (латинський, російський і український).

- altKey − повертає true, якщо була натиснута клавіша "Alt", і false у противному випадку (тільки для події keyDown);
- altLeft − повертає true, якщо була натиснута ліва клавіша "Alt", і false у противному випадку (тільки для події keyDown);
- ctrlKey − повертає true, якщо була натиснута клавіша Ctrl, і false у противному випадку (тільки для події keyDown);
- ctrlLeft − повертає true, якщо була натиснута ліва клавіша Ctrl, і false у противному випадку (тільки для події keyDown);

- shiftKey − повертає true, якщо була натиснута клавіша "Shift", і false у противному випадку (тільки для події keyDown);
- shifLeft − повертає true, якщо була натиснута ліва клавіша "Shift", і false у противному випадку (тільки для події keyDown);
- repeat − повертає true, якщо подія keyDown настала повторно внаслідок того, що користувач втримав клавішу натиснутою, і false у іншому випадку.

Хоча при натисканні правих керуючих клавіш ("Alt", "Ctrl" і "Shift") об'єкт event не формує властивостей, аналогічних тим, які створюються для лівих управляющіз клавіш, визначити, чи була натиснута права керуюча клавіша, можна таким чином: якщо вираз event.altKey&&!event.altLeft повертає значення true, значить була натиснута саме права клавіша "Alt".

Особливості роботи з клавіатурою показані у ***прикладі 13.3***, де шляхом обробки подій, пов'язаних з натисненням певних клавіш, виконуються завантаження рисунків на Web-сторінці, їх переміщення по екрану і зміна розміру.

Приклад 13.3

```
<HTML>
<HEAD>
<TITLE> Робота з клавіатурою </TITLE>
<SCRIPT>
function keyEvent()
{
status='Unicode =' + event.keyCode + 'Символ =' +
String.fromCharCode(event.keyCode);
if ((event.keyCode> = 49)&&(event.keyCode <= 51))
{
imgs.width = 150;
imgs.style.position = "absolute";
imgs.style.left = 0;
imgs.style.top = 0;
}
switch(event.keyCode)//Аналіз Unicode клавіш
{
case 37: (imgs.style.pixelLeft-= 10; break)//"Стрілка
вліво"
```

```
     case 38: (imgs.style.pixelTop-= 10; break)//"Стрілка
вниз"
     case 39: (imgs.style.pixelLeft + = 10; break)//"Стрілка
вправо"
     case 40: (imgs.style.pixelTop + = 10; break)//"Стрілка
вгору"
     case 49: (imgs.src = "flower.jpg"; break)//"1"
     case 50: (imgs.src = "everest.jpg"; break)//"2"
     case 51: (imgs.src = "fish.gif"; break)//"3"
     case 188: (imgs.width--; break;)//"<"
     case 190: (imgs.width + +; break;)//">"
     }
     }
     function help()
     {
     alert("Вибір рисунка: клавіші '1','2' і '3'\n" +
     "Переміщення рисунка: клавіші 'Стрілка ліворуч','Стрілка
вправо'," +
     "'Стрілка вниз' і 'Стрілка вгору'\n" +
     "Зміна розміру рисунка: клавіші '<' і '>'");
     event.returnValue=false;
     }
     </SCRIPT>
     </HEAD>
     <BODY OnHelp="help()" onKeyDown="keyEvent()">
     <IMG NAME=imgs WIDTH=0>
     </BODY>
     </HTML>
```

Після завантаження Web-сторінки з'являється порожнє вікно браузера. При натисканні будь-якої клавіші клавіатури по події keyDown викликається функція keyEvent (), яка аналізує Unicode клавіш і виконує наступні дії:

- виводить у поле статусу вікна браузера значення Unicode клавіші і її символ, отриманий методом fromCharCode() об'єкта String;

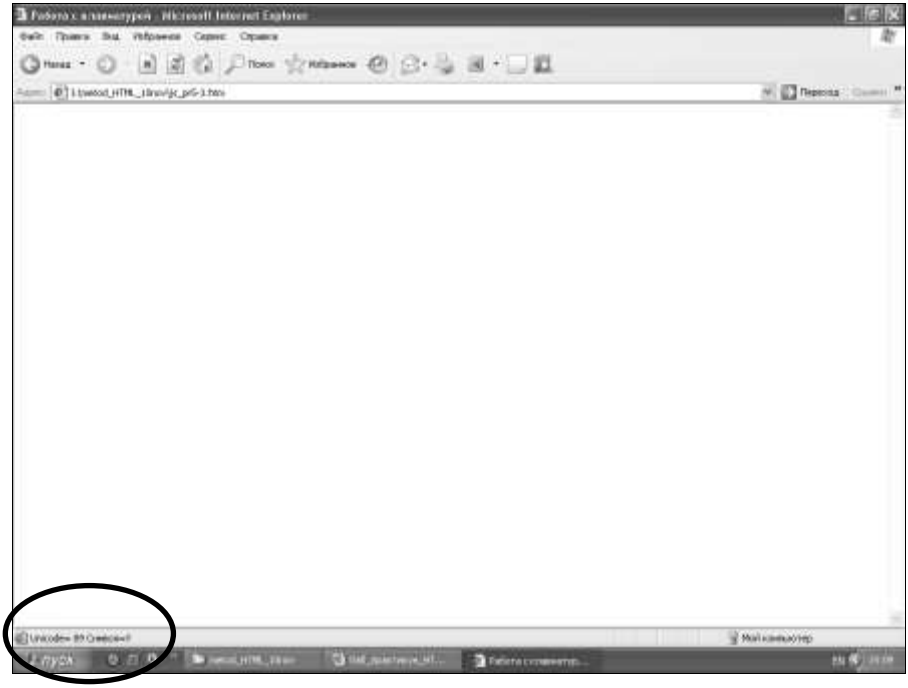


Рисунок 13.3 – результат приклада 3 на екрані

- по натисненню клавіш "1", "2" і "3" - виводить у вікно браузера зображення відповідно квітки, гори Еверест і рибок;
- при натисканні клавіш "Стрілка вліво", "Стрілка вправо", "Стрілка вниз" та "Стрілка вгору" − переміщує отримане зображення по вікну браузера;
- при натисканні клавіш ">" і "<" − збільшує або зменшує розмір зображення.

При натисканні клавіші "F1" по події Help викликається функція Help(), яка виводить на екран підказку про призначення клавіш, які використовуються в прикладі. Оскільки ця подія обробляється також браузером, можна заборонити висновок повідомлень браузера, вказавши у функції значення властивості returnValue об'єкта event, рівним false. Це також можна зробити дещо по іншому:

```
<BODY onHelp="help(); retorn false"
```

Індивідуальні завдання

- Розробити Web-сторінку, яка містить трьохшарову структуру, кожен шар якої складається з тегів, зазначених у колонці "Теги шарів" таблиці 13.1.
- За подіями 1, 2, 3 і 4, які полягають у одночасному натисканні двох клавіш (перша вказана у колонці "Буква", друга − одна з клавіш "1", "2", "3" і "4") виконати наступні дії:
  - по події 1 показати тільки перший шар;
  - по події 2 показати тільки другий шар;
  - по події 3 показати тільки третій шар;
  - по події 4 показати одночасно два шари, номери яких вказані в колонці "Шари".
- Використовуючи властивість CSS display:
  - по події 5 − помістити у вікно браузера коментар або пояснення, що стосується одного з елементів Web-сторінки;
  - по події 6 − видалити виведені коментар або пояснення. Примітка: якщо подія 6 співпадає з подією 5, це означає, що вона полягає у повторному виконанні події 5.
- При натисканні клавіші "F1" вивести на екран підказку (help) про призначення використовуючих клавіш.

Таблиця 13.1 – Варіанти індивідуальних завдань

№	Теги шарів			Шари	Буква	Події	
	1	2	3			5	6
1	IMG	P, P	INPUT, INPUT	1, 2	ю	лівий Alt	лівий Alt
2	P, SPAN	IMG	P, FONT	2, 3	Ф	лівий Alt	лівий Ctrl
3	IMG	A, A	TABLE	1, 2	ї	лівий Alt	лівий Shift
4	OL	IMG	TEXTAREA	2,3	Ж	лівий Ctrl	лівий Alt
5	IMG	P, A	TABLE	1, 3	ц	лівий Ctrl	лівий Ctrl
6	IFRAME	IMG	UL	2, 3	щ	лівий Ctrl	лівий Shift
7	IMG	SELECT	P, B	1, 3	Л	лівий Shift	лівий Alt
8	IMG	P, I, B	INPUT/ Radio	1, 2	ь	лівий Shift	лівий Ctrl
9	IMG	IMG	INPUT/ checkbox	1, 3	Ч	лівий Shift	лівий Shif
10	IMG	P, P	INPUT/reset	1, 2	И	лівий Alt	правий Alt
11	TABLE	UL	SPAN, A	5	ї	лівий Alt	правий Ctrl
12	P, A	IMG	IFRAME	1, 2	Ї	лівий Alt	правий Shift
13	H1	TABLE	OL	1, 3	є	лівий Ctrl	правий Alt
14	H2, P	EMBED	IFRAME	2, 3	Є	лівий Ctrl	правий Ctrl
15	EMBED	IMG	P, SPAN	1, 2	q	лівий Ctrl	правий Shift
16	UL	EMBED	OL	2, 3	z	лівий Shift	правий Alt
17	EMBED	P, A, A	IMG	1, 2	w	лівий Shift	правий Ctrl
18	H2, SPAN	EMBED	INPUT/text	2, 3	y	лівий Shift	правий Shif
19	IMG	P, P	TEXTAREA	1, 2	ю	правий Alt	лівий Alt
20	P, U	IMG	SPAN, FONT	2, 3	Ф	правий Alt	лівий Ctrl

ЛАБОРАТОРНА РОБОТА 14
КЛАС ДАТИ DATE

Мета: Вивчення методів класу Date для роботи з датами і тимчасовими інтервалами на мові JavaScript.

14.1. Створення об'єктів класу Date

Клас дати Date служить для зберігання значень дати й часу. Об'єкти цього класу створюються за допомогою конструктора Date ([par]), який приймає значення дати у числовому або рядковому форматі.

Якщо параметр par, представлений у числовому форматі, він трактується як число мілісекунд, які минули з півночі 1 січня 1970 за Гринвічем. Якщо він представлений у послідовному форматі, конструктор намагається представити його у значенні дати або часу за такими правилами:

- рядок, який має формат "місяць/число/рік" або "месяц.число.год" (наприклад, 04.30.10) перетворюється у дату;
- рядок, який має формат "місяць число рік" (наприклад, April 30 2010) перетворюється у дату;
- рядок, який має формат "години: хвилини: секунди" (наприклад, 10:04:57) перетвориться у час;
- рядок, який має формат "години: хвилини PM" (наприклад, 10:04 PM) перетвориться у час.

Наприклад, дата запуску першого штучного супутника землі може бути задана таким чином:

```
sputnikLaunch = new Date("October квітня, 1957 19:28:34 GMT");
```

Конструктор об'єкта дати може мати і такий формат: Date (Рік, Місяць, Число [, Годинники [, Хвилини [, Секунди [, Мілісекунди]]]]). Якщо не один параметр не заданий, конструктор ініціалізує об'єкт поточною датою.

Приклади ініціалізації об'єктів дати наведені у ***прикладі 14.1*** та ***прикладі 14.2***.

14.2. Методи класу Date

Об'єкти класу Date використовують наступні методи:

- getTime() − повертає ціле число мілісекунд, які минули з півночі
- 1 січня 1970 за Гринвічем;
- getDay() − повертає ціле число, яке позначає день тижня: 0 - неділя, 1 - понеділок і т.д.;
- getDate() − повертає день місяця (ціле число від 1 до 31);
- getMonth() − повертає ціле число, яке вказує номер місяця (від 0 до 11);
- getFullYear () − повертає рік;
- getYear()− повертає рік (застосовується для сумісності, рекомендується використовувати getFullYear());
- getHours() − повертає годину (ціле число від 0 до 59);
- getMinutes() − повертає хвилини (ціле число від 0 до 59);
- getSeconds() − повертає секунди (ціле число від 0 до 59);
- getMilliseconds () − повертає мілісекунди (ціле число від 0 до 999);
- getTimezoneOffset() − повертає різницю у хвилинах між локальним та універсальним часом UTC *;
- getUTCTime() − повертає число по UTC;
- getUTCDay() − повертає число по UTC;
- getUTCDate() − повертає число по UTC;
- getUTCMonth() − повертає ціле число по UTC;
- getUTCFullYear() − повертає рік по UTC;
- getUTCHours() − повертає годину по UTC;
- getUTCMinutes() − повертає хвилини за UTC;
- getUTCSeconds() − повертає секунди за UTC;
- getUTCMilliseconds() − повертає мілісекунди за UTC;
- setTime(Число) − встановлює час з урахуванням числа мілісекунд, які минули з півночі 1 січня 1970 за Гринвічем, заданих у якості параметра;
- setDate(Число) − встановлює день місяця;
- setMonth(Місяць [, Число]) − встановлює місяць і число, якщо задано за UTC;

- setFullYear(Рік [, Місяць [, Число]]) − встановлює рік, а також місяць і число якщо задані за UTC;
- setYear(Рік) − встановлює рік (застосовується для сумісності, рекомендується використовувати setFullYear();
- setHours(Година [, Хвилини [, Секунди [, Міллісевкунди]]]]) − встановлює годину, а також хвилини, секунди і мілісекунди, якщо задан за UTC;
- setMinutes(Хвилини [, Секунди [, Міллісевкунди]]]) − встановлює хвилини, а також секунди і мілісекунди, якщо задані за UTC;
- setSeconds(Секунди [, Міллісевкунди]) − встановлює секунди, а також мілісекунди, якщо задані;
- setMilliseconds(Міллісевкунди) − встановлює мілісекунди;
- setUTCDate(Число) − встановлює день місяця за UTC;
- setUTCMonth(Місяць [, Число]) − встановлює місяць і число, якщо задано за UTC;
- setUTCFullYear(Рік [, Місяць [, Число]]) − встановлює рік, а також місяць і число, якщо задані за UTC;
- setUTCHours(Година [, Хвилини [, Секунди [, Міллісевкунди]]]]) − встановлює годину, а також хвилини, секунди і мілісекунди, якщо задані за UTC;
- setUTCMinutes(Хвилини [, Секунди [, Міллісевкунди]]]) − встановлює хвилини, а також секунди і мілісекунди, якщо задані за UTC;
- setUTCSeconds(Секунди [, Міллісевкунди]) − встановлює секунди, а також мілісекунди, якщо задані за UTC;
- setUTCMilliseconds(Міллісевкунди) − встановлює мілісекунди за UTC;
- parse − розшифровує рядок згідно з наведеними вище правилам і повертає число мілісекунд, які пройшли між отриманою датою і північчю 1 січня 1970 за Гринвічем;
- toUTCString() − перетворює дату у рядок у форматі універсального часу і возвраащает її;
- toGMTString() − перетворює дату у рядок у форматі Гринвіческому часу і возвраащает її (застосовується для сумісності, рекомендується використовувати toUTCString();

- tolocaleString() − перетворює дату у рядок, використовуючи інтернаціональні установки системи, і возвраащает її (не рекомендується використовувати для обчислень);
- toSource − повертає рядок, який представляє вихідний код дати;
- toString () − перетворює дату у рядок і возвраащает її.

Примітка: Універсальний час UTC (Universal Coordinated Time) було введено замість часу за Гринвічем GMT (Greenwich Mean Time) через нерівномірні шкали GMT, які пов'язані з нерівномірним обертанням Землі. Універсальний час визначає середній сонячний час на меридіані Грінвіча на базі рівномірної шкали атомного часу. UTC не змінюється ні взимку, ні влітку.

Використання методів класу Date показано у ***прикладі 14.1*** та ***прикладі 14.2***.

У ***прикладі 14.1*** визначається число днів між сьогоднішнім днем і датою наступного Різдва (у прикладі − 7 січня 2011 року). Для цього після ініціалізації двох об'єктів Date, які відповідають цим датам, визначається число мілісекунд, що містяться в цьому проміжку часу, який потім приводиться до числа днів.

Приклад 14.1

```
<HTML>
<HEAD>
<TITLE> Визначення кількості днів до Різдва </TITLE>
<SCRIPT>
today = new Date();
nextXmas = new Date("January 7, 2011");
msPerDay = 24*60*60*1000; // Число мілісекунд у добі
daysLeft = (nextXmas.getTime()-today.getTime()) /
msPerDay;
daysLeft = Math.round(daysLeft);
alert("До наступного різдва залишилося" + daysLeft +
"днів");
</SCRIPT>
</HEAD>
</HTML>
```

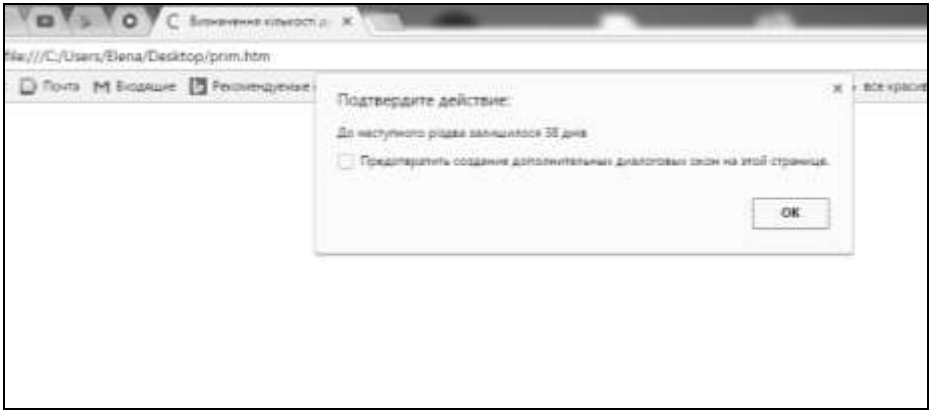


Рисунок 14.1 – результат приклада 14.1 на екрані

У ***прикладі 14.2*** здійснюється виведення значень поточної дати і поточного часу у вікно браузера, а також виконується задана тимчасова затримка.

Приклад 14.2

```
<HTML>
<HEAD>
<TITLE> Висновок поточної дати і часу на екран </TITLE>
<STYLE>
text (font:8mm; color:blue; position:absolute; left:500;
top:5; background:#E0E0FF;)
</STYLE>
<SCRIPT>
function outTime()
{
txt.className = "text";
txt.style.display = "";
cur_time = new Date();
with(cur_time)
txt.value = getDate() + "." + eval(getMonth()+1) + "." +
getYear()+ "" +
getHours()+":"+ getMinutes()+":"+ getSeconds();
setTimeout ("outTime()", 50);
}
```

```
      function delay()
      {
      d = delay.arguments [0];
      if (delay.arguments.lengh == 1) par = 0;
      else par = delay.arguments [1];
      switch(par)
      {
      case 0: break;
      case 1: d *= 1000; break;
      case 2: d *= 60000; break;
      }
      {
      status = "Виконується затримка";
      base_time = new Date().getTime();
      test = new Date(); alert(test.getTimezoneOffset());
      while(new Date(). getTime()-base_time <d);
      status = "";
      }
      }
      </SCRIPT>
      </HEAD>
      <BODY OnClick = "outTime()" onContextmenu = "delay(5,1);
return false"">
      <INPUT TYPE=text NAME="txt" SIZE=7 STYLE="display:none">
      </BODY>
      </HTML>
```

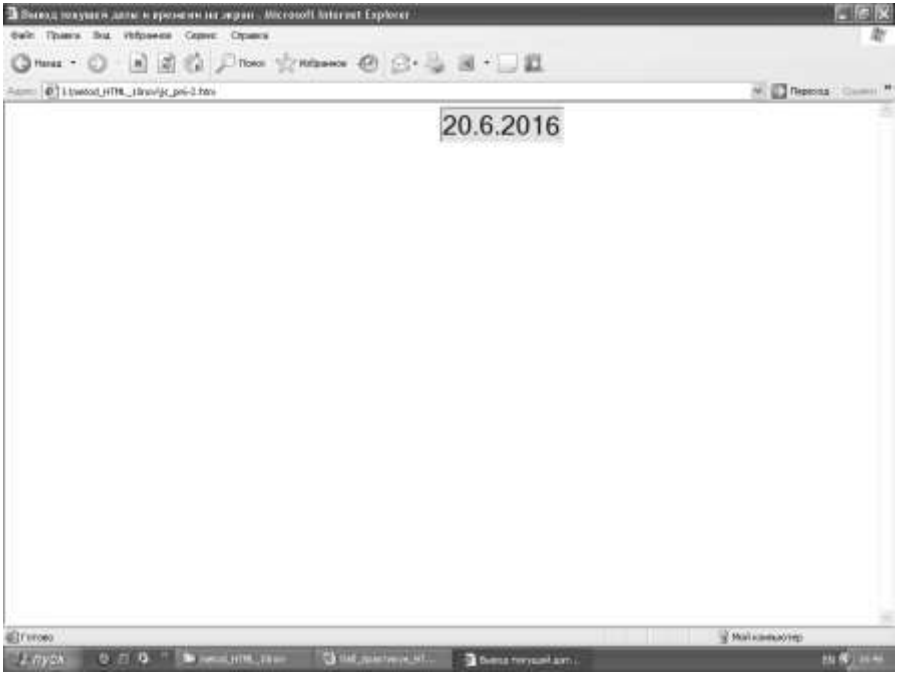


Рисунок 14.2 – Результата приклада 14.2 на екрані

Подією Click викликається функція outTime(), яка встановлює необхідні значення властивостей текстового поля тега <INPUT> і виводить у це поле поточну дату і поточний час, які визначені з використанням методів класу Date. Розмір поля та шрифту були вибрані так, щоб після активізації поля можна було б, натискаючи клавіші "End" і "Home", показувати у текстовому полі по черзі або дату, або час.

Подією Contextmenu виконується виклик функції Delay(5,1), яка здійснює п'ятисекунду затримку шляхом порівняння поточного часу з базовим, певним на початку затримки. Протягом затримки у полі статусу вікна браузера виводиться повідомлення "Виконується затримка".

Особливістю цієї функції є те, що з двох її параметрів перший є обов'язковим − він вказує чисельне значення затримки, а другий, який визначає, в яких одиницях задається затримка, є необов'язковим. У разі, якщо він не вказаний при виконанні функції, його значення приймається за замовчуванням. Другий параметр може приймати такі значення:

- 0 − час затримки вказується у мілісекундах (значення за умовчанням);
- 1 − час затримки вказується у секундах;
- 2 − час затримки вказується у хвилинах.

Щоб визначити, скільки параметрів було передано функції у кожному конкретному випадку та їх значення, необхідно скористатися масивом arguments, який, будучи властивістю класу функцій Function, містить значення всіх аргументів, переданих функції, а: висловлення delay.arguments [0] задає перший аргумент функції, вираз delay.arguments [1] − другий аргумент, а delay.arguments.length − кількість аргументів, переданих функції. Зауважимо, що масив arguments доступний тільки у тілі функції.

Індивідуальні завдання

Використовуючи HTML, CSS і JavaScript виконати наступне:

- Створити дві кнопки: "Місцевий час" і "UCT". При натисканні будь-якої з кнопок виводити на екран відповідне динамічно змінюється час (ГГ: ММ: СС) і утримувати його на екрані протягом однієї хвилини.
- Здійснити введення двох довільних дат і визначити число днів між ними.
- При натисканні клавіш клавіатури "ч" "м" і "з" виводити на екран відповідно години, хвилини і секунди поточного часу.
- З масиву рядків випадковим чином виводити на екран слово, вводити це слово на клавіатурі і визначати середній час введення одного символу.
- Ввести рік і визначити число неділь для кожного місяця цього року.
- Визначити, наскільки швидкість введення російського тексту вище швидкісті введення латинського тексту (або навпаки).
- Визначити число вихідних днів у році − неділь і свят (свята задати окремим масивом).
- Ввести час (ГГ: ММ: СС) і після натискання комбінації клавіш ctrl-T вивести на екран одне з повідомлень: "До певного часу залишається ... хвилин" або "Після заданого часу минуло ... хвилин".

- Вивести текст на екран і визначити середній час виведення одного символу.
- Ввести довільну дату і визначити дату через 100 днів.
- Завантажити одночасно кілька рисунків на екран і визначити середній час завантаження одного рисунка.
- По натисненню будь-якої клавіші клавіатури виводити у верхній лівий кут вікна браузера поточну дату і час і утримувати на екрані протягом 30 секунд.
- Вивести питання на екран, прийняти відповідь і визначити, чи не перевищує цей час відповіді заданий час.
- На екран випадковим чином виводити по одній літері російського алфавіту. Після виведення кожної букви вводити цей символ з клавіатури і визначити букву, час введення якої з клавіатури максимальний.
- Порівняти швидкодію арифметичних операцій на JavaScript з цілими та числами із плаваючою точкою.
- По будь-якої події, пов’язаної з роботою мишкою, виводити на екран у полі статусу поточний дінамично мінливий час (ГГ: ММ: СС) і утримувати на екрані 20 секунд.
- Визначити, скільки разів за останні 10 років п'ятниця припадала на 13 число місяця.
- Ввести дату дня народження і визначити усі роки, коли цей день потрапляв на неділю.
- Визначити усі дати, коли читалися лекції за дисципліною "Проектування Web-сайтів".
- На найближчі 5 років визначити, на який день тижня припадає Новий рік (31 грудня).

СПИСОК ЛІТЕРАТУРИ

1. К. Дари, Б. Бринзаре, Ф. ЧЕРЧЕЗ-Тоза, М. Бусика. "AJAX и PHP: Разработка динамических веб-приложений". – СПб.: Плюс, 2006. – 336 с.

2. Дронов. В.А. JavaScript в Web-дизайне. – Спб.: БХВ-Петербург, 2005. – 880 с.

3. Фролов А.В., Фролов Г.В. Сервер Web своими руками. Язык HTML, приложения GGI и ISAPI, установка Web-серверов. – М.: Диалог-Мифи, 1997. – 288 с. - (Библиотека современного программиста; т. 29).

4. Ломов А.Ю. HTML, CSS, скрипты: практика создания сайтов. – СПб.: БХВ–Петербург, 2007. – 416 с.

5. Петюшкин А.В. HTML в Web-дизайне. – СПб.: БХВ-Петербург, 2005. – 400 с.

6. Полонская Е.Л. Язык HTML. Самоучитель. – М.: Диалектика, 2005. – 320 с.

7. Холл Матри, БраунЛэрри. Программирование для Web. Библиотека программиста. – М.: Издательский дом «Вильямс», 2002. – 1264 с.

8. Кингсли-Хью Э., Кингсли-Хью К. JavaScript 1.5: Учебный курс. – СПб: Питер, 2001. – 272 с.

9. Чебыкин Р. Самоучитель HTML и CSS. Современные технологии. – М.: БХВ-Петербург, 2008. – 109.

10. М. П. Левин, Ю. М. Алексеев. Самоучитель разработки Web-сайтов: HTML, CSS, графика, анимация. – М. : Триумф, 2007. – 400 с.

ЗМІСТ

Навчальне видання

ЗИКОВ Ігор Семенович
ЧЕРНИХ Олена Петрівна

Web-проектування

Практикум
для студентів комп'ютерних спеціальностей

Роботу до видання рекомендував *В.Д. Дмитрієнко*

Редактор *Н.В.Верстюк*

План 2015 р., поз. 85

Підп. до друку __.__.__. Формат 60×84 1/16. Папір друк. №2.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 11,6. Наклад. 100 прим.
Зам №_____. Ціна договірна.

Видавничий центр НТУ „ХПІ".
Свідоцтво про державну реєстрацію ДК № 116 від 10.07.2004 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ "ХПІ". 61002, Харків, вул. Фрунзе, 21